Engineering PFLOTRAN for Scalable Performance on Cray XT and IBM BlueGene Architectures

Richard Tran Mills[1], **Vamsi Sripathi**[2], **G. (Kumar) Mahinthakumar**[3], **Glenn E. Hammond**[4], **Peter C. Lichtner**[5], **Barry F. Smith**[6]

[1]Computational Earth Sciences Group, Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6015

[2]Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206

[3]Department of Civil, Construction, and Environmental Engineering, North Carolina State University, Raleigh, NC 27695-7908

[4]Hydrology Group, Environmental Technology Division, Pacific Northwest National Laboratory, Richland, WA 99352

[5]Hydrology, Geochemistry, and Geology Group, Earth and Environmental Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545

[6]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4844

E-mail: rmills@ornl.gov, vamsi_s@ncsu.edu, gmkumar@ncsu.edu, glenn.hammond@pnl.gov, lichtner@lanl.gov, bsmith@mcs.anl.gov

Abstract. We describe PFLOTRAN—a code for simulation of coupled hydro-thermal-chemical processes in variably saturated, non-isothermal, porous media—and the approaches we have employed to obtain scalable performance on some of the largest scale supercomputers in the world. We present detailed analyses of I/O and solver performance on Jaguar, the Cray XT5 at Oak Ridge National Laboratory, and Intrepid, the IBM BlueGene/P at Argonne National Laboratory, that have guided our choice of algorithms.

1. Introduction

Subsurface (groundwater) flow and reactive transport models have become important tools for tasks such as understanding contaminant migration and cleanup, assessing geothermal energy technologies, and exploring geologic carbon sequestration as a possible means to reduce greenhouse gas emissions. Although simple groundwater models will suffice in some instances, there there are many in which sophisticated, three-dimensional, multi-scale models employing multiple fluid phases and chemical components coupled through a suite of biological and geochemical reactions are required. The increased complexity of such models demands computing power far beyond that of desktop computers, and in some cases requires the ability to utilize true leadership-class supercomputers.

In this paper, we present analyses of I/O and solver performance in one such complex groundwater simulation code, PFLOTRAN, on two of the most powerful computers in the world—Jaguar, the Cray XT5 at Oak Ridge National Laboratory, and Intrepid, the IBM BlueGene/P at Argonne National Laboratory—and discuss some of the strategies we have employed to achieve parallel scalability on these machines.

2. The subsurface reacting flow simulator, PFLOTRAN

PFLOTRAN [1–6] solves a coupled system of mass and energy conservation equations for a number of phases, including air, water, black oil, and supercritical CO_2, and for multiple chemical components. It utilizes a finite-volume spatial discretization combined with either operator-split or backward-Euler (fully implicit) time stepping. PFLOTRAN is written in Fortran 95 using as modular and object-oriented an approach as possible within the constraints of the language standard, and, being a relatively new code, it is unencumbered by legacy code has been designed from day one with parallel scalability in mind. It is built on top of the PETSc framework [7–9] and makes extensive use of features from PETSc, including iterative nonliner and linear solvers, distributed linear algebra data structures, parallel constructs for representing PDEs on structured grids, performance logging, runtime control of solver and other options, and binary I/O. It employs parallel HDF5 [10] for I/O and SAMRAI [11] for adaptive mesh refinement.

PFLOTRAN employs domain-decomposition parallelism, with each subdomain assigned to an MPI process and a parallel solve implemented over all processes. A number of different solver and preconditioner combinations from PETSc or other packages can be used, but in this paper we employ an outer, inexact Newton method with an inner BiCGStab linear solver or an "improved" variant (described in Section 3.1), preconditioned with a block-Jacobi method employing point-block ILU(0) on each subdomain. Message passing (3D "halo exchange") is required to exchange ghost points across subdomain boundaries, and, within the BiCGStab solver, gather/scatter operations are needed to handle off-processor vector elements in matrix-vector product computations, and global reduction operations are required to compute vector inner products and norms.

3. Scalability on Cray XT and IBM BlueGene Architectures

PFLOTRAN exhibits good parallel scalability on a number of different machine architectures including Jaguar, the Cray XT5 at ORNL, and Intrepid, the IBM BlueGene/P at Argonne, and has been run on up to 131,072 compute cores. Figure 2 illustrates the strong-scaling performance (using the default BiCGStab implementation) observed on the Cray XT5 and the IBM BG/P for the flow and reactive transport solves for a benchmark problem that simulates the release of a hypothetical uranium plume at the Hanford 300 Area in southeastern Washington state. This problem is based on that described in [12]. Our version of the benchmark problem consists of $850 \times 1000 \times 80$ cells and includes 15 primary chemical species, resulting in 68 million degrees of freedom for the flow solve and approximately one billion degrees of freedom for the reactive transport solve (flow and transport are coupled sequentially in this simulation).

Speedup tapers off on the XT5 primarily due to the high cost of allreduce operations on the machine. Figure 1 illustrates the scaling of the time spent in MPI_Allreduce() for the combined benchmark flow and transport problem; the BG/P outperforms the XT5 at 16380 processor cores, which coincides closely with the cross-over point for the flow solve in Figure 2. Unlike the BG/P, the XT5 does not have a separate, tree-structured network for such operations, which partly explains the poorer scalability on the XT5. We believe that operating system noise is also a significant contributor to the poor scalability observed on the XT5, as we have observed significant load imbalance on synthetic MPI_Allreduce() benchmarks that are perfectly load balanced within the application. Figure 3 displays box plots (also known as box-and-whisker plots) summarizing the MPI_Allreduce() timings (including synchronization time) on the Cray XT5 and IBM BG/P. Interpreting these plots is somewhat less than straightforward because

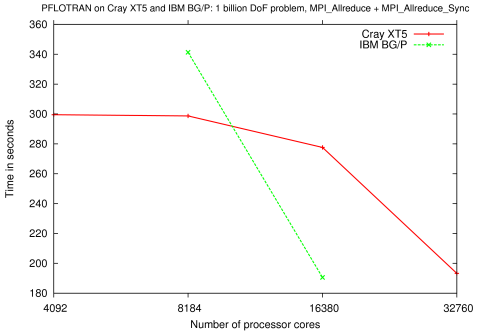

Figure 1. Scaling of time spent in MPI_Allreduce() for the benchmark flow and transport problem.

the simulation employs a structured grid containing inactive cells along the river boundary; MPI processes with many inactive cells have very little computation to do, and therefore spend a long time waiting at MPI_Allreduce() calls. These lightly-loaded processes are responsible for the great majority of the outliers (represented by "+" symbols in the plots), although they constitute only about 5% of the processes. Note that these wait times are much shorter on the XT5 because the CPUs are faster, and not because the communication is better.

3.1. Mitigating Cost of Global Reductions in Krylov Solvers

At high core counts (on the Cray XT5, especially), linear solves may be dominated by cost of MPI_Allreduce() calls to calculate vector dot products and norms. We have added a PETSc implementation of the so-called Improved BiCGStab algorithm (IBCGS) [13] that requires only 2 MPI_Allreduce() calls per iteration, instead of the usual 3. If the residual norm calculation is lagged by an iteration, it is possible to wrap everything into a single MPI_Allreduce(), at the cost of doing one additional iteration; we lag this calculation in the flow solve, where a high number of iterations are required, but not in the transport solve, where few linear solver iterations are required at each Newton step. The algorithm is considerably more complicated, requires the transpose matrix vector product (applied only during the first iteration), and performs some extra, local vector operations.

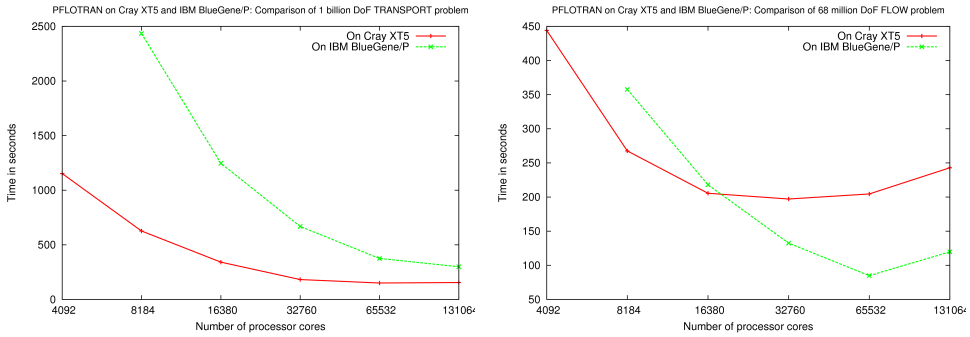

Figure 2. Scaling on the Cray XT5 and IBM BG/P systems for 30 steps of the uranium plume migration problem benchmark. The reactive transport solve (left) has 1 billion degrees of freedom, and the corresponding flow problem (right) has 68 million degrees of freedom. Note that the flow problem is very small for these core counts.

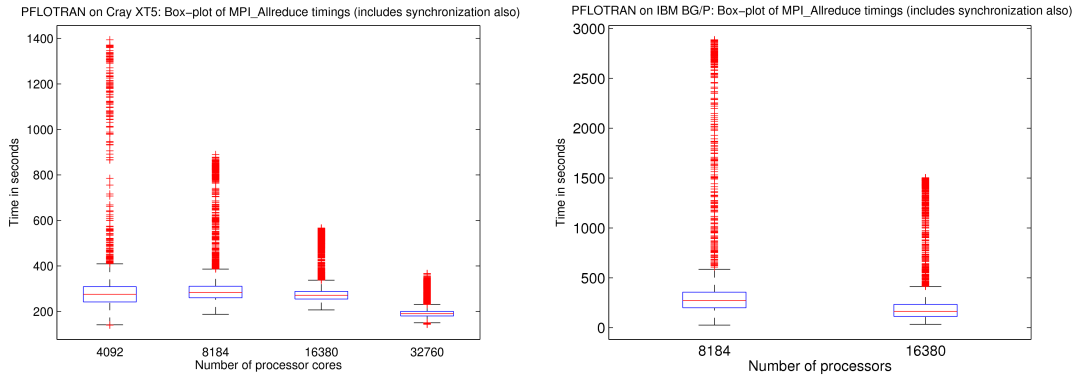

Figure 3. Box-plots of MPI_Allreduce timings (including synchronization time) for the XT5 (left) and BG/P (right) for the flow and reactive transport benchmark problem. Points which cross the whiskers are marked with "+" symbols and represent outliers.

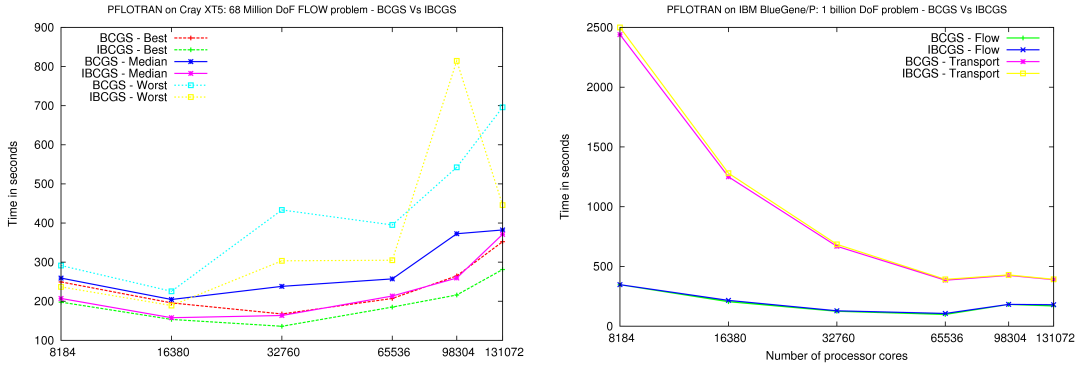

Figure 4. Comparisons of the conventional BiCGStab (BCGS) algorithm and the Improved (IBCGS) algorithm for a 68 million degrees of freedom flow problem on the Cray XT5 (left) and IBM BG/P (right, with performance for accompanying transport problem with 1 billion degrees of freedom also depicted). This is an artificially small problem for these high core counts, so the cost of communication dominates, as there is relatively little computation per core to be done. Due to variability in machine performance as well as IBCGS iteration counts, best, worst, and median times are reported for the XT5 runs.

Despite the somewhat higher operation count, the reduced communication costs can lead to significant performance gains; we have found that the restructured solver is generally faster on the Cray XT4/5 using more than a few hundred processor cores. Figure 4 displays the performance of the conventional and restructured BiCGStab solvers on the Cray XT5 and IBM BG/P. Due to instability within the restructured solver, the number of iterations required to solve the same problem varies between runs. The restructed algorithm offers significant performance improvements on the Cray XT5, but on the IBM BG/P, where global reduction operations exhibit better scalability, we observe little performance improvement.

3.2. Improving Parallel I/O with a Two Phase Approach

PFLOTRAN uses parallel I/O in the form of 1) routines employing parallel HDF5 to read input files and write out simulation output fies, and 2) direct MPI-IO calls in a PETSc Viewer backend to write out checkpoint files. The original HDF5 routines perform well on the IBM BG/P architecture, but on the Cray XT architecture, these routines do not scale to high core counts due to too many small I/O requests, and contention for the small number of Lustre Metadata Servers. The rememdy this problem, members of the SciDAC Performance Engineering Research Institute (G. Mahinthakumar and V. Sripathi) have collaborated with the PFLOTRAN team to implement a two-phase (communication phase and I/O phase) approach [14]. Instead of all processes participating in collective read or write operations, the MPI global communicator is split into sub-communicators, with the root process of each communicator performing I/O for the entire group and appropriate gather/scatters to collect or distribute data to/from group members. We note that MPI-IO implementations generally provide support for two-phase I/O, but we have chosen to implement this at the application level because 1) In the PFLOTRAN initialization phase, not all processes participate in all HDF5 read calls, and 2) Attempting to use the MPI-IO two-phase I/O results in exhaustion of resources in the low-level Portals library used for inter-node communication on the Cray XT architecture. Our improved I/O routines yield **25X improvement** in the initialization phase and **3X improvement** in the write phase at 65,536 cores (quad-core processors) on the Cray XT5.

4. Conclusions and Future Work

The performance of PFLOTRAN is being continuously benchmarked and improved as the code is developed and new machines come online. In this paper we have examined the

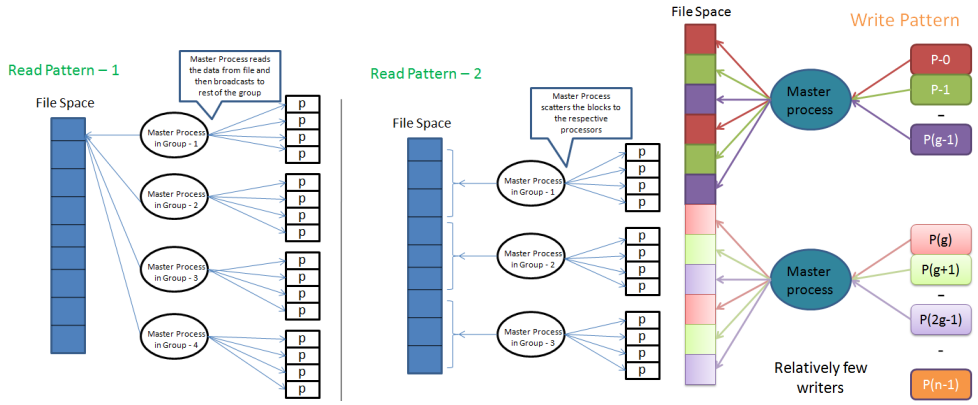

Figure 5. Improved (two phase) read and write patterns. There are two different patterns of reads in the initialization phase.

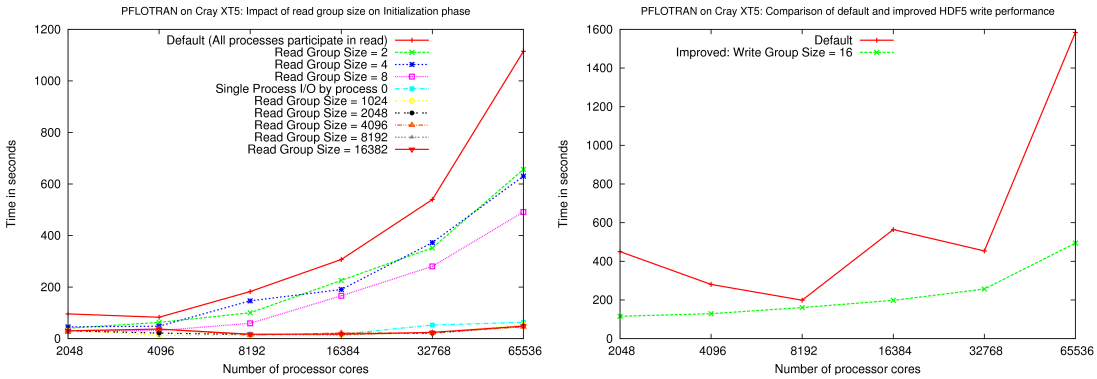

Figure 6. Comparisons of default initialization (read) and write behavior in PFLOTRAN with improved two phase implementations.

performance of PFLOTRAN on two leadership-class machine architectures—the Cray XT5 and IBM BlueGene/P—and discussed algorithmic changes made to solvers (restructured BiCGStab algorithm) and I/O routines (two-phase I/O) to address particular performance bottlenecks we have observed. Such work will continue as as new capabilities (e.g., unstructured grids, structured adaptive mesh refinement, multi-continuum formulations) are developed and as new architectures (e.g., nodes with GPU or other "accelerator" devices) come online. At a more fundamental level than performance tuning, we also will continue to investigate scalable solver and preconditioner algorithms that are well-tailored to our application while also possessing intrinsic scalability.

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 Energy 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.

References

[1] Mills R T, Lu C, Lichtner P C and Hammond G E 2007 Simulating subsurface flow and transport on ultrascale computers using PFLOTRAN *SciDAC 2007 Scientific Discovery through Advanced Computing* (*Journal of Physics: Conference Series* vol 78) ed Keyes D (Boston, Massachusetts: IOP Publishing) p 012051

[2] Hammond G E, Lichtner P C and Lu C 2007 Subsurface multiphase flow and multicomponent reactive transpo rt modeling using high-performance computing *SciDAC 2007 Scientific Discovery through Advanced Computing* (*Journal of Physics: Conference Series* vol 78) ed Keyes D (Boston, Massachusetts: IOP Publishing) p 012025

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

[4] Hammond G E, Lichtner P C, Mills R T and Lu C 2008 Towards petascale computing in geosciences: application to the Hanford 300 Area *SciDAC 2008 Scientific Discovery through Advanced Computing* (*Journal of Physics: Conference Series* vol 125) ed Keyes D (Seattle, Washington: IOP Publishing) p 012051

[5] Mills R T, Hammond G E, Lichtner P C, Sripathi V, Mahinthakumar G K and Smith B F 2009 Modeling subsurface reactive flows using leadership-class computing *SciDAC 2009 Scientific Discovery through Advanced Computing* (*Journal of Physics: Conference Series* vol 180)

[6] Hammond G E, Lichtner P C, Mills R T and Lu C 2010 *Ground Water Reactive Transport Models* (Bentham Science Publishers) chap PFLOTRAN: Reactive Flow & Transport Code for Use on Laptops to Leadership-Class Supercomputers

[7] Balay S, Buschelman K, Gropp W D, Kaushik D, Knepley M G, McInnes L C, Smith B F and Zhang H 2009 PETSc Web page http://www.mcs.anl.gov/petsc

[8] Balay S, Buschelman K, Eijkhout V, Gropp W D, Kaushik D, Knepley M G, McInnes L C, Smith B F and Zhang H 2009 PETSc users manual Tech. Rep. ANL-95/11 - Revision 3.0.0 Argonne National Laboratory

[9] Balay S, Gropp W D, McInnes L C and Smith B F 1997 Efficient management of parallelism in object oriented numerical software libraries *Modern Software Tools in Scientific Computing* ed Arge E, Bruaset A M and Langtangen H P (Birkhäuser Press) pp 163–202

[10] The HDF Group 2009 *HDF5 User's Guide: HDF5 Release 1.8.3* NCSA http://www.hdfgroup.org/HDF5

[11] Wissink A M, Hornung R D, Kohn S R, Smith S S and Elliott N 2001 Large scale parallel structured AMR calculations using the SAMRAI framework *Supercomputing '01: Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)* (New York, NY, USA: ACM) pp 6–6 ISBN 1-58113-293-X

[12] Hammond G E and Lichtner P C 2010 Cleaning up the Cold War: Simulating uranium migration at the Hanford 300 Area *Proceedings of SciDAC 2010*

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

[14] Sripathi V 2010 *Performance Analysis and Optimization of Parallel I/O in a large scale Groundwater Application on Petascale Architectures* Master's thesis North Carolina State University