GRID-enabled Solution of Groundwater Inverse Problems on the TeraGrid Network

Mohamed Sayeed, Kumar Mahinthakumar and Nicholas T. Karonis

SIMULATION 2007 83: 437
DOI: 10.1177/0037549707084936

GRID-enabled Solution of Groundwater Inverse Problems on the TeraGrid Network

Mohamed Sayeed
Purdue University
msayeed@purdue.edu

Kumar Mahinthakumar
North Carolina State University
gmkumar@ncsu.edu

Nicholas T. Karonis
Northern Illinois University/Argonne National Laboratory
karonis@niu.edu

Grid-computing environments are becoming increasingly popular for scientific computing due to the significant increase in capacity that they represent when compared to a single computational resource (e.g., a single cluster), their ubiquitous availability and advances in grid middleware components. Several such specialized grid environments are now available for users in the commercial and research sectors. One such effort for research is the TeraGrid, consisting of a collection of geographically distributed heterogeneous supercomputer resources including data storage resources. Parallel implementations for these environments are inherently multilevel and obtaining efficient mapping of work to processors can be extremely challenging. This paper extends an existing MPI application to the grid via the use of grid-enabled MPI libraries. The application uses a simulation-optimization framework involving coarse-grained parallelism in the optimizer and fine-grained parallelism in the finite-element-based simulator. Using parallelism at both these levels is essential for problems involving computationally intensive simulation steps. A hierarchical grid architecture consisting of a collection of supercomputers is ideally suited for these types of problems as a good application-to-architecture mapping can be obtained with a proper implementation. This paper presents the performance results of our implementation on the TeraGrid network consisting of three geographically distributed supercomputers.

Keywords: High performance computing, grid computing, inverse problems, optimization, groundwater, genetic algorithms, finite element methods, TeraGrid

1. Introduction

Groundwater characterization problems often require solution of an *inverse problem* as the measured data are generally sparse and contain only the signatures of the desired information. The desired information has then to be extracted from the observed data by solving the inverse problem. Solution to the inverse problem typically requires solution to the *forward problem*. In many cases, the model describing the forward problem (forward model) is a partial differential equation (PDE) system that describes a forward relationship between the desired information and the measured data. In this paper, the forward model is a three-dimensional PDE system that describes multicomponent groundwater transport.

Inverse problems are often *ill-posed*, implying that the inverse solution may not exist (non-existence), more than one solution may exist (non-uniqueness) or the solutions may be too sensitive to measurement errors (instability) [48]. In practice, the typically large number of potential solutions for an inverse problem makes enumeration of potential solutions impractical.

If the problem is linear then direct inversion methods based on matrix solution methods can be used to solve

SIMULATION, Vol. 83, Issue 6, June 2007 437–448
© 2007 The Society for Modeling and Simulation International
DOI: 10.1177/0037549707084936
Figures 1, 3–7 appear in color online: http://sim.sagepub.com

the inverse problem Sun, N.-Z. 1994. *Inverse Problems in Groundwater Modeling, Theory and Applications of Transport in Porous Media*. Kluwer Academic Publishers, 6,337 pp. Use of indirect optimization-based methods is an attractive alternative to solving inverse problems due to their greater flexibility than direct methods and computational efficiency when compared to enumeration. In this case, a mathematical or heuristic optimization technique is used to search the space of potential solutions efficiently for the solution that most closely approximates observed values (i.e., minimizes the error between the observed and calculated values). Regardless of the method used, however, solution of inverse problems can be several orders of magnitude more *computationally challenging* than solution of the corresponding forward problem since hundreds-to-thousands of solutions of the forward model are typically required.

The ill-posed nature of many environmental inverse problems, combined with the computational requirements inherent in solving many thousands of forward problems, often render the application of optimization approaches intractable, limiting current solution practices to trial-and-error or oversimplification of the problem's complexities [43]. With recent emergence of *grid computing* [14], use of optimization approaches is becoming increasingly feasible for the solution of a wide range of environmental inverse problems. The goal of this paper is to develop a grid-enabled simulation-optimization (S/O) framework to solve groundwater inverse problems.

1.1 Grid Computing

Computational Grids enable the coupled and coordinated use of geographically distributed resources for such purposes as large-scale computation, distributed data analysis and remote visualization that would not otherwise be tractable or sometimes even possible on any single resource. The development or adaptation of applications for grid environments is made challenging, however, by the often heterogeneous nature of the resources involved and the fact that these resources typically reside in different administrative domains, run different software, are subject to different access control policies and may be connected by networks with widely varying performance characteristics.

The past decade has produced a number of grid middleware tools including Globus [16], Legion [19], Condor-G [8], and Netsolve [7] to name a few of the most popular. Core components typically define a protocol for interacting with a remote source plus an application program interface (API) used to invoke that protocol. Higher-level libraries, services, tools and applications use core services to implement more complex global functionality. To this end, a number of problem solving environments (PSE) have been developed for grid applications during this time (Yang *et al*. [54], Prashar *et al*. [42] and Schreiber *et al*. [46]). These PSEs provide grid services, setting up (job submission and scheduling of computational resources), steering and monitoring of simulations, and in some cases the analysis packages as well. Not coincidentally over that same period we have seen the emergence of several national and continental-level grids around the world. The US National Science Foundation's TeraGrid (TG) [51], the US Department of Energy's Earth System Grid (ESG) [52], Open Science Grid (OSG) [53], tri-lab DISCOM grid [50], DARPA's agent grid [9], Japan's National Research Grid Initiative (NAREGI) [23] and the European Commission project Enabling Grids for E-sciencE (EGEE) [13] are just a few examples of the many grids in existence today.

Most grid applications demonstrated to date typically fall under one of the following three categories: (a) embarrassingly parallel applications (e.g., parameter search), (b) functional pipelines or work flows or (c) single jobs (typically simulations) that marshal the resources of multiple sites to form a single virtual computational resource. Applications in the first category launch a large number of nearly independent computations on the grid. Many grid applications demonstrated to date fall under this category (e.g., [32], [28], [35], [27] and [29]). The use of grid computing for these types of applications (while justified in some cases) is not always warranted since traditional distributed computing using clusters of workstations can be used equally well. The second category, functional pipelines, usually referred to as work flows, split jobs by assigning functional components to the facilities that are best suited to the subtask. For example, a job may start with data mining where there is a large data repository, results from that mining may then be shipped to a large computational resource for intensive computation whose results are, in turn, sent to a third facility with state-of-the-art scientific visualization equipment for visualization (e.g., [19] and [11]). Grid applications in the third category are the most challenging and typically involve scientific and engineering applications with problem sizes of interest that require resources that exceed the capacity (most often memory) of even the largest single computing facility. Despite the challenges posed by these types of applications, computational grids have been successfully used in cross-site simulations to solve record-setting and award-winning problems across a broad spectrum of scientific and engineering domains (e.g., [2], [6], [10], [11], [12] and [22]). Most of these successful demonstrations have relied heavily on significant tuning of both application (e.g., overlapping communication and computation, alternate decompositions, etc.) and grid middleware (alternate communication protocols, polling frequency, etc.) and dedicated grid resources.

The simulation-optimization application developed in this paper can be imagined as a hybrid of the first and third category. At a coarse-grained level, it is somewhat embarrassingly parallel; however, at a fine-grained level it is tightly coupled and communication intensive. The computational grid architecture consisting of geographically distributed supercomputers maps suitably for these types of applications where a large number of tightly coupled parallel simulations (each simulation using a moderate number of processors on a tightly coupled machine) need to be performed simultaneously. However, challenges need to be overcome, such as the periodic synchronization and communication at the coarse-grained level (requiring multi-site coordination and load balancing) and enforcing locality of all fine-grained activity.

The rest of the paper is organized as follows. In section 2 we describe our S/O framework and section 3 discusses its grid-implementation. In section 4 we describe the test problem setup and present our results in section 5. Finally we conclude in section 6. Some of the results presented are from data collected in 2003 when the system was still in friendly user mode (Table 2, Figures 6 and 7). Unless otherwise specified, all results are from 2006.

2. Algorithmic Framework

Our S/O framework consists of two major components: (a) a genetic algorithm (GA) based optimizer and (b) a parallel finite-element (FEM) simulator. In this implementation these components are tightly coupled to produce a single parallel MPI (Message Passing Interface [20]) executable.

2.1 GA Optimizer

Our optimizer consists of genetic algorithms (GAs) and a variety of gradient-based and non-gradient-based local search approaches. However, the simulations performed in this study are limited to real encoded GAs. Here we provide a brief discussion of GAs for completeness. GAs optimize using a search process that emulates natural evolution. In a GA, a potential solution to a problem is represented as a vector (chromosome string), which can consist of binary or real values. The GA starts with a set of potential solutions or population, which is often generated at random. The performance of each solution is characterized by a fitness value. In the context of inverse problems, fitness is calculated using a forward solve and is inversely proportional to the difference between computed and observed values. During the GA search process, the population is iteratively subjected to several probabilistic operators that are analogous to natural selection, mating (including genetic recombination) and mutation. Each iteration through these steps constitutes a generation. Because fitter solutions are more likely to be selected for mating, the incidence of good traits of a solution tends to increase from one generation to the next. Crossover serves to sample these traits in many different combinations. Typically, the quality of solutions improves quickly at the onset of the algorithm. As the population converges, improvements diminish. Often, a stopping criterion is used to determine when improvements are sufficiently small that additional computations are not warranted. These concepts are well described in many texts, including [17] and [36].

In the groundwater test problems considered here, the objective function (f) to be minimized is the root square error (RSE) between the observed and the computed output concentration signals at a few selected observation points in the domain:

$$f = RSE = \sqrt{\sum_{i=1}^{n}(C_i^{obs} - C_i^{calc})^2}, \qquad (1)$$

where C_i^{obs} is the observed concentration and C_i^{calc} is the calculated concentration at the observation points, i is the observation number and n is the total number of observations. For the hypothetical cases tested in this paper, the C_i^{obs} values are synthetically generated. In order to compute the output signals C_i^{calc} for each individual in a GA operation, a forward groundwater transport simulation is performed. Because the computer-intensive fitness calculations for each individual in a generation can proceed independently, GAs are amenable for use in a parallel or distributed computing environment.

2.2 FEM Simulator

The parallel transport simulator PGREM3D [33] employed in the objective function evaluations solves the multi-component groundwater transport problem. The general system of equations describing transport of nc dissolved components undergoing reactions in saturated media is defined by

$$\begin{aligned}\frac{\partial C_i}{\partial t} &= \nabla \cdot (\mathbf{D}\nabla C_i) - \nabla \cdot (C_i\mathbf{v}) + \frac{q}{\theta}(C_i - C_{0i}) - R_i \\ i &= 1, 2, 3..., nc, \end{aligned} \qquad (2)$$

where $\mathbf{v}$ is the 3×1 velocity field vector, $\mathbf{D}$ is the 3×3 dispersion tensor dependent on $\mathbf{v}$ and C_i is the dissolved concentration of component i. The term $q(C_i\text{-}C_{0i})/\theta$ represents the source term with volumetric flux q, medium porosity θ and injected concentration C_{0i} (e.g., from injection wells). R_i is the rate of mass loss of component i due to sorption, bioremediation and ion exchange reactions and is the main coupling term for the system of equations. The term R_i may contain many terms and can be nonlinear [4].

The system of transport and reaction equations is discretized using the Galerkin finite element method (FEM) with 8-noded linear hexahedral elements. A logically rectangular grid structure is assumed but irregular geometries are supported using distorted elements. A Crank-Nicolson approximation (central finite-difference) is used for the time derivative terms. A lumped mass formulation [21] is used for all time-derivative and non-derivative (zeroth

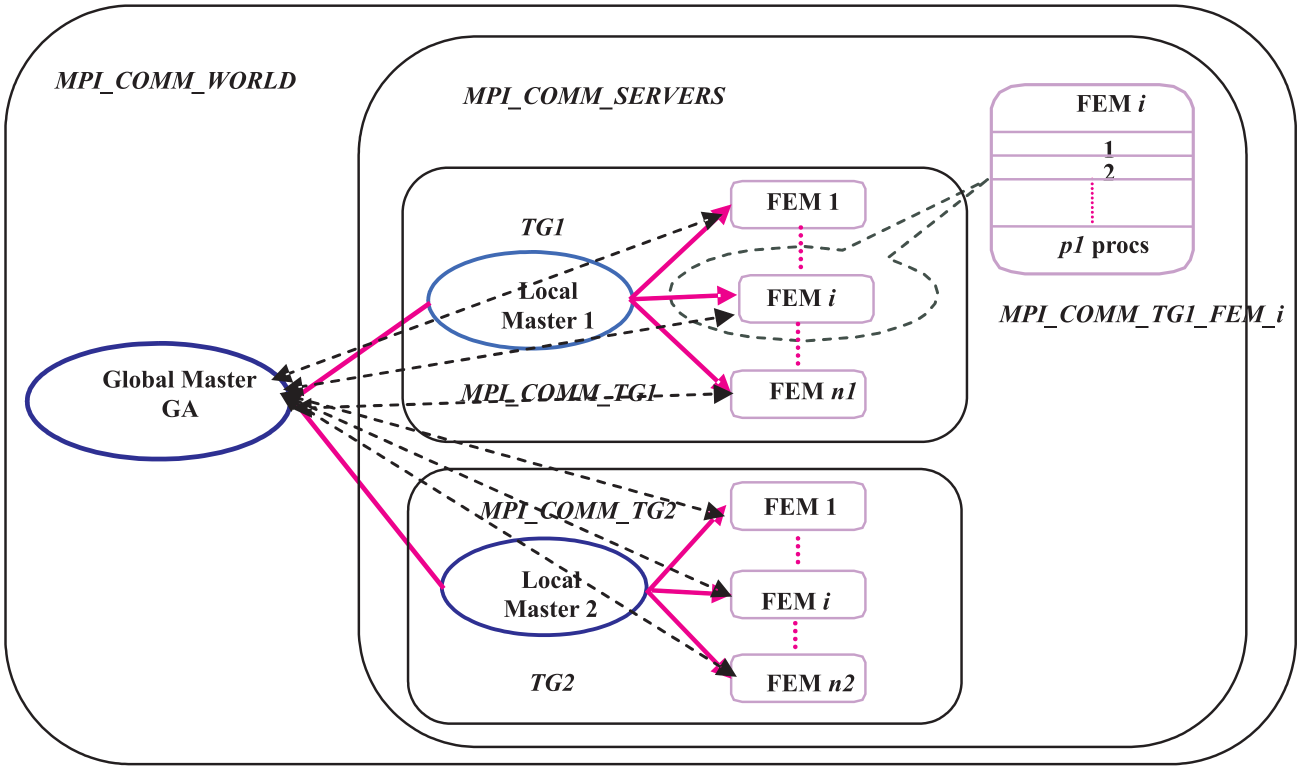

Figure 1. Schematic layout of the parallel GA-FEM simulation-optimization framework.

spatial derivative) terms. The coupled nonlinear system is solved using a modified form of the Sequential Iterative Algorithm (SIA). Several Krylov subspace iterative solvers are implemented in the code for the matrix solution [34]. Simulations in this paper are conducted using the BiCGStab solver, which performs reasonably well for most problems.

The transport simulator is parallelized using a two-dimensional domain decomposition (in the x and y directions) using asynchronous (non-blocking) message passing (MPI library) to exchange information between these domains. The simulator has been tested extensively for scalability and performance on a variety of parallel architectures. Details regarding the simulator can be found in the following articles: flow simulator [44] and transport simulator [31], [30].

3. Grid Implementation

The implementation uses a three-tier communication hierarchy, and uses communicators at specific levels for reading input files and broadcasting them to other processes at that level, thereby reducing costly I/O time. A self-scheduling algorithm keeps all the server processes in a group busy; however, at the end of each generation/cycle the processes are synchronized. A restart option is also available for the GA to restart its operations from where it stopped. The sections that follow describe the parallel implementation of the optimization framework for a single supercomputer and its extension to the grid environment.

The processes and groups corresponding to our implementation are shown schematically in Figure 1. For GAs the task parallelism is limited to the number of individuals in the GA population. The communication can be carried out at three levels between the client process and server processes. Let us assume we have a total of $N + 1$ processes for our coupled GA-FEM simulation, and we have 2 TeraGrid (TG) sites (TG1 and TG2), each with N1 and N2 processes available such that N1 + N2 = N + 1. At first, all the $N + 1$ processes are assigned to the world group with the default communicator MPI_COMM_WORLD (level-0). Server subgroups n1 and n2 are created local to TG sites TG1 and TG2 respectively, such that $N1 = n1 * p$, $N2 = n2 * p$, where p is the number of processes used for one FEM simulation. The challenge here is to make sure that the processes in the TG1 or TG2 subgroups are local to one site, as there is no explicit control in MPI-1 to achieve this. To address this we use the MPI-

1 call "MPI_Get_processor_name" to get the processor's name and then check its locality during TG1 or TG2 subgroup formations. MPICH-G2 supplied communicator attributes MPICHX_TOPOLOGY_DEPTHS and MPICHX_TOPOLOGY_COLORS now provide the same functionality to group the processes based on their location; however, we did not use these as these features were released after our original implementation. These lowest level groups (e.g., MPI_COMM_TG1_FEM_i) are called server subgroups and each will perform one FEM simulation at a time.

At each level, local process ID numbers (or ranks) will be assigned to each process in each subgroup. The process with subgroup rank 0 will serve as the subgroup leader. The process with rank 0 in MPI_COMM_WORLD is the "Global Master" that performs the GA computations and also manages the server processes. The global master can be one of the processes at a TG site or an independent process on a client workstation. In our case this process is at one of the TG sites (NCSA). Since the basic input file is the same for all the server subgroups performing the transport simulations, only one process from each TG site, in our case the process with rank 0 at level-1, will need to read the input file. Once read, the input data are broadcast to all the other server processes using the level-1 group communicator. Using this mechanism avoids the need for each server process to read the input file, thus preventing I/O conflicts and also possibly saving on costly I/O time. All local communication within each transport simulation is handled using the level-2 communicator.

The global master sends the string representing the decision variables directly (dashed arrows in Figure 1) to the server subgroup leaders (i.e., processes with rank 0 at level-2). These are then broadcast to others in the same group. The subgroup processes will perform the transport run and the server subgroup leader will return the RSE value to the global master using the world group communicator.

To achieve reasonable load balancing on the TeraGrid for these operations, we use a centralized self-scheduling strategy that is commonly used in master-slave applications [18]. At the beginning of each GA generation, the decision variables are dispatched in a round-robin fashion to all the server subgroup leaders. Assuming that the number of server subgroups is smaller than the GA population size, the next decision variable set will be sent to the group that last returned an RSE value. This process is repeated until all individuals in a population of a generation are evaluated. However, the global master needs to synchronize the processes at the end of each generation.

In this work, we have used two different grid-enabled MPI libraries: MPICH-VMI, which uses VMI (Virtual Machine Interface) [41], and MPICH-G2 [25], [38], which is based on the Globus Toolkit [16]. Additional information on these libraries is presented in section 5.

4. Test Problem

The groundwater inverse problem solved in this study is the release history reconstruction problem. Reconstructing the release histories of contaminant sources is important in identifying responsible parties in a groundwater contamination incident and for allocating remedial costs [37]. A three-source release history reconstruction problem is solved as a test case. A schematic of the problem is shown in Figure 2. The observation data are synthetically generated using a forward transport simulation with assumed arbitrary release histories and then corrupted with a random white noise. The releases are modeled using internal time-varying first-type (i.e., prescribed concentration) boundary conditions at the source locations. Initial conditions are assumed to be the release concentrations during the first time period at the sources. The release history is reconstructed for 10 time periods. The third source is assumed to be a dummy source (no release). The locations of all the potential sources are assumed to be known, as is typically the case in real world situations. The unknown decision variables are the release concentrations at each time period for each source. For example, the three-source problem will have 30 decision variables. The problem uses a fully heterogeneous flow field and is therefore more realistic.

The 3D domain is of size 500 m $\times$ 300 m $\times$ 10 m (51 $\times$ 31 $\times$ 11 grid) and the time-step size is 20 days. Simulations are performed for 1000 time steps (20 000 days). The sources are assumed to be active for the first 10 000 days. There are 10 time periods, with each time period equaling 1000 days. A velocity of 0.1 m/day implies that a release occurring at the upstream end will take approximately 5000 days (13.5 years) to travel the entire domain. These and other parameters used in the problem are shown in Table 1. A total of 18 sampling points (observation wells) with 9 each distributed uniformly at the two-thirds and downstream vertical cross sections (*yz* planes) are utilized (Figure 2). At each sampling point, observations are recorded once every 50 time steps. Therefore, a total of 360 observations are used corresponding to the 18 sampling points and 1000 time steps. As an example, the synthetically generated plume after 6000 days for a three-source problem is shown in Figure 3 at the mid-horizontal *xy* plane. To account for measurement and model errors, a 10% random white noise error is applied to the synthetically generated "observation" data.

4.1 Prediction Efficiency

The prediction efficiency of the RGA-FEM framework for this problem is shown in Figure 4. S1, S2 and S3 denote sources 1, 2 and 3 respectively. The results indicate that while RGA captures the approximate release history it is nevertheless a very crude approximation. Prediction efficiency is particularly poor for the dummy source (S3).

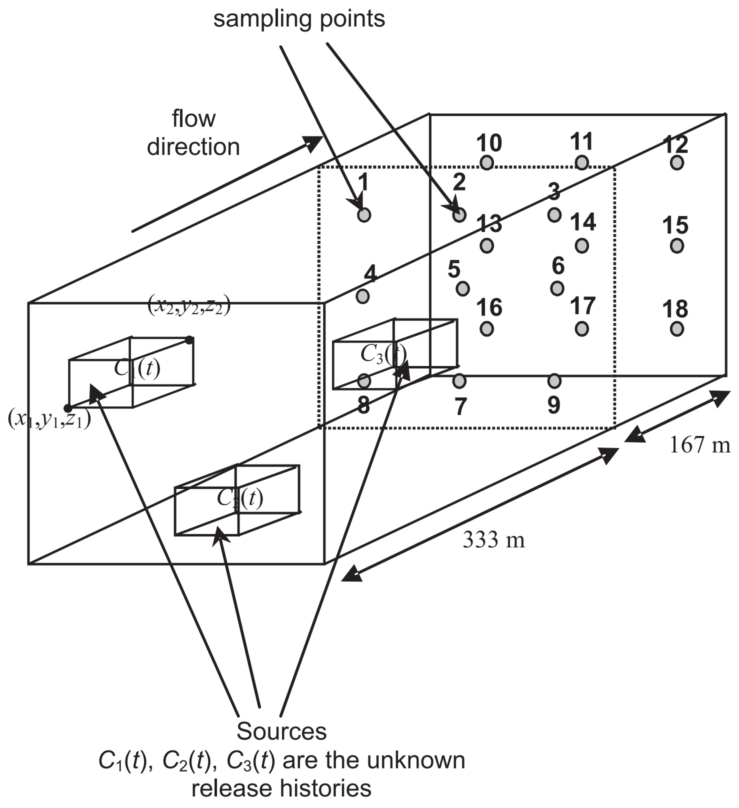

Figure 2. Layout of the 3-sources release history test problem.

Table 1. Input parameters for three-source release history problem.

Parameter	**Values**
Problem size	51×31×11
Number of time steps	1000
Time step size (Δt)	20 days
Grid spacing	$\Delta x = 10$ m, $\Delta y = 10$ m, $\Delta z = 1$ m
Dispersion Parameters	$\alpha_L = 1$ m, $\alpha_{TH} = 0.1$ m, $\alpha_{TV} = 0.01$ m, $D_m = 0.001$ m^2/d
Hydraulic conductivity field	Heterogeneous with a mean of 1 m/d
Velocity field	Variable in space with a mean of 0.1 m/d
Source Locations (three-source) (x_1, y_1, z_1) (x_2, y_2, z_2)	(8,9,3) (10,11,5) (18,19,2) (20,21,4) (35,8,7) (37,10,9)

This indicates that while RGA is very good for obtaining an initial approximation, it is not very efficient in fine-tuning the results. To obtain a more accurate estimate we will need to use this solution as a starting guess for a local search (LS) algorithm. This is called a hybrid GA-LS approach. Although we have successfully used this method for solving these kinds of problems [45], we refrain from showing these results in this paper to keep our focus to a pure GA-FEM framework.

5. Grid Computing Experiments

In this paper grid computing is enabled by the grid middleware toolkit Globus, MPICH-G2 and MPICH-VMI. The MPICH-G2 library is a grid enabled implementation of MPI v1.1 standard and uses some of the services provided by Globus Toolkit (e.g., job startup, security, etc). In order to grid enable any MPI application the user has to link with these libraries and use the provided run-time environ-

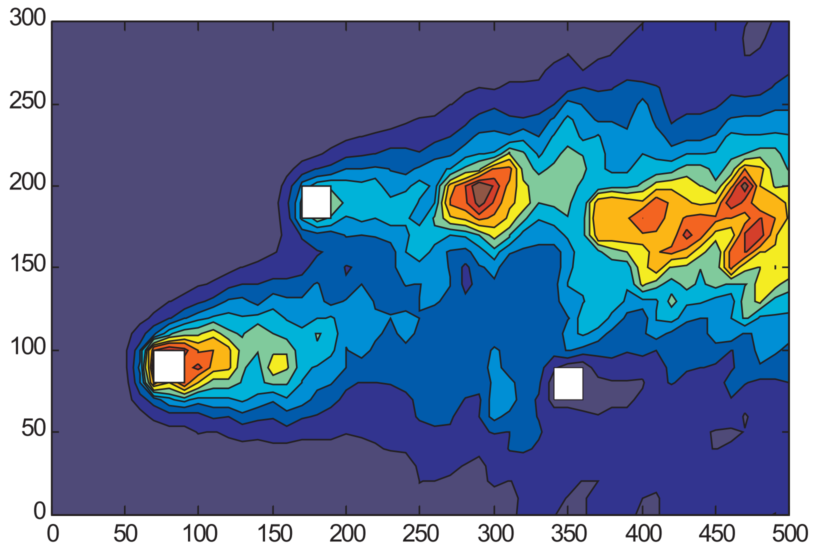

Figure 3. Horizontal cross-section of the synthetic plume.

ments for job execution. MPICH-G2 uses Globus tools for remote job submission, scheduling and monitoring. For experiments reported in this paper using MPICH-G2 we used the TG's installation of MPICH v1.2.6e, which was built using v4.0.3 of the Globus Toolkit. The MPICH-G2 library uses MPICH-GM (Myrinet) library underneath for message passing within a single site and uses TCP for cross-site message passing. The other grid enabled MPI, MPICH-VMI, is based on the virtual machine interface middleware communication layer. These libraries allow data striping across heterogeneous networks and architectures and use multi-protocol communication. This paper reports results from both MPICH-G2 and MPICH-VMI. Most of the VMI experiments were performed in August 2003 and used VMI-1.1. Recent VMI experiments used VMI-2.1. Most of the MPICH-G2 experiments were performed more recently (2006). The TeraGrid common software stack supports only MPICH-G2, although older installations of VMI-2.1 are available only at NCSA (National Center for Supercomputer Applications) and SDSC (San Diego Supercomputer Center). Hence recent experiments using VMI are only between these sites. Additional information on MPICH-G2 and VMI is available online at [38] and [39], respectively.

The experiments are carried out at the SDSC, NCSA and UC/ANL (University of Chicago and Argonne National Laboratory) TeraGrid sites. The TeraGrid Linux clusters are similar systems with dual processors Itanium2 nodes using Myrinet interconnect. The hardware and software features of the TG machines are available online at [51]. As mentioned earlier the coarse-grained task parallelism and fine-grained data parallelism makes this application ideal for the grid environment. Since the application has two levels of communication hierarchy (one

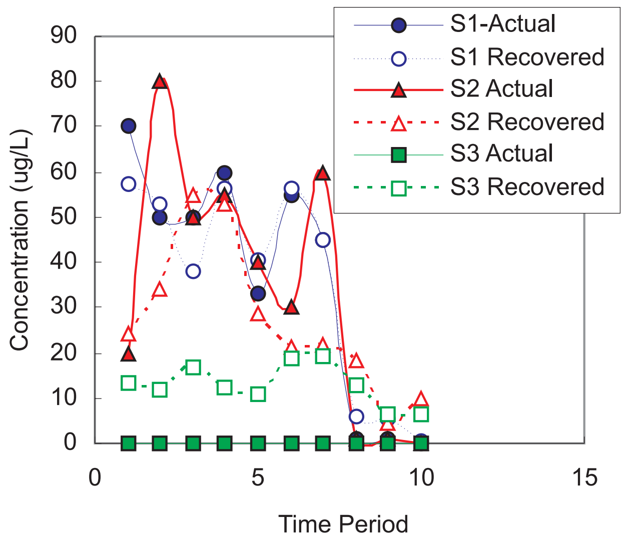

Figure 4. Prediction accuracy of RGA.

loosely coupled and the other tightly coupled) it is analogous to the hardware characteristics of the grid.

The first step was to port the GA-FEM code to the NCSA, SDSC and UC/ANL TeraGrid Linux clusters. It primarily required minor compilation error fixes. Extensive single site runs and a limited number of cross-sites were done to evaluate performance. Recall that our GA-FEM implementation has both fine and coarse-grained levels of parallelism. The fine-grained parallelism (FEM transport simulator) computations are communication intensive. A simple MPI profiling study of the standalone

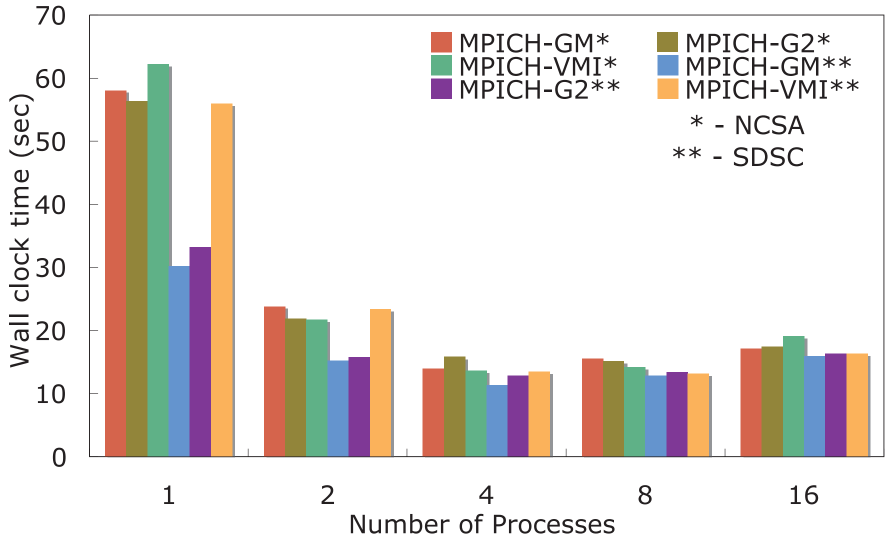

Figure 5. Single site fine-grained FEM parallelism using different MPI flavors.

FEM code revealed a communication overhead of 20% for a four-processes run on a single cluster. Thus for cross-site runs, the processes forming the FEM subgroups are ensured to be local to a single site and are not across sites. The cross-site runs were difficult to perform during the friendly user mode as the resources at the two sites had to be available at the same time and synchronized. The synchronization step was to submit the Globus RSL script or VMI script at the multiple sites almost simultaneously and hope for the jobs to start concurrently. New runs used cross-site reservation to help co-scheduling of jobs.

We measured cross-site (NCSA and SDSC) interconnect/network latency and bandwidth using MPICH-VMI (2003) and MPICH-G2 (2006) by sending and receiving messages of varying sizes between two compute processes with MPI_Send/MPI_Recv calls (i.e., ping-pong). The test provides information about the latencies for small messages and the actual available bandwidth for large size messages. Network latencies and bandwidths measured using the Fortran version of the ping-pong program revealed varying performance. Since the application is latency bound (message sizes are typically in the range of hundreds of bytes – about 30 doubles at a time for a three-source release history problem), we focus on latencies rather than bandwidth. The measured roundtrip latencies ranged between 60 ms and 200 ms. The higher latencies were observed for runs conducted recently (in 2006) as the TeraGrid was in production mode with increased network traffic and switch contention. We note here that the reported round trip time (RTT) as measured and reported by the TeraGrid [49] between NCSA and SDSC is approximately 30 ms. The inter-site network performance reported on [49] is collected from special hardware (i.e., not compute nodes) at each site. The higher numbers we report are possibly due to software overhead (MPI + Fortran) and the fact that the messages are sent between compute nodes incurring potential switch contention. Thus, for cross-site runs proper care must be taken to reduce the number of small messages sent across and to bundle the messages if necessary. Also latency-hiding mechanisms such as overlapping computations with communication should be considered to further improve performance.

5.1 FEM and RGA-FEM performance on TeraGrid

The performance of standalone FEM code (fine-grained performance) is compared in this section for single-site and cross-site runs. Figure 5 shows the performance of the standalone FEM application at different TG-sites (single site runs) using different installed MPI libraries. The performance of the grid enabled libraries (MPICH-G2 and MPICH-VMI) is comparable to the vendor supplied MPICH-GM library with a little overhead. Also, for these single site runs we observe super-linear speedup behavior going from 1 to 2 processes. This is likely due to improved cache behavior when solving smaller problems per process. Communication and synchronization costs start to dominate beyond four processes, resulting in no or very little speedup. Communication costs in the FEM code are dominated by the large number of MPI_Allreduce calls involved in the dot product operations in the BiCGStab linear solver. Although the TeraGrid Linux clusters at these sites (NCSA and SDSC) are very similar, we see significant single processor performance difference (see Figure 5). This may be because of initial system configuration issues.

While one would not normally perform fine-grained computations across TG sites, we performed some ex-

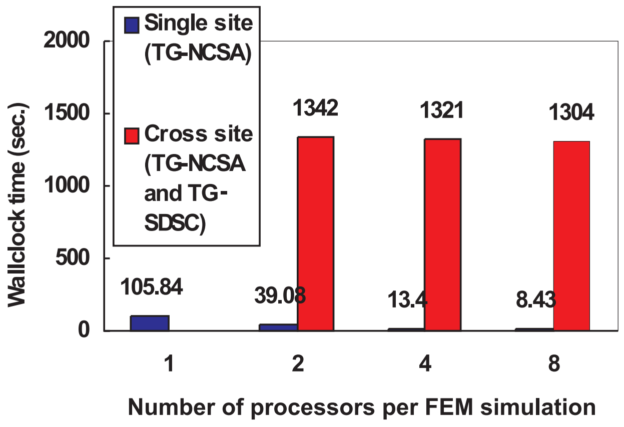

Figure 6. Single and cross-site fine-grained parallelism using MPICH-VMI.

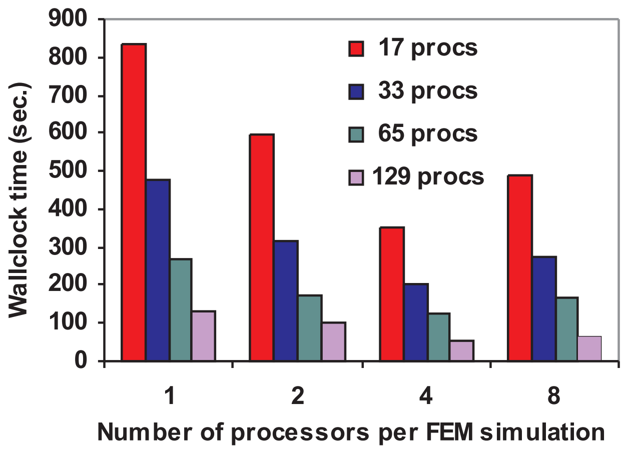

Figure 7. Effect of fine-grained parallelism on RGA-FEM performance.

periments to show the adverse impact of performing such computations. Figure 6 illustrates this behavior using MPICH-VMI. As expected, cross-site fine-grained computations show a very large communication overhead. This is mainly due to the high network latency between TG-NCSA and TG-SDSC. For example, for the 8 processes per FEM simulation case, single cluster run takes only 8 seconds whereas the cross-site run takes approximately 1,300 seconds. This clearly highlights the importance of performing the fine-grained FEM simulations locally on a single cluster.

Figure 7 shows the effect of fine-grained parallelism in the overall RGA-FEM performance within a single site (NCSA) using MPICH-VMI. The results show that the best performance is achieved when using four processes for each FEM simulation and this is consistent with the

Table 2. Runtimes for the RGA-FEM simulations using MPICH-VMI1

Number of processes for RGA-FEM simulation	Number of processes per FEM simulation			
	1	2	4	8
Single site (NCSA) run time (sec) using VMI1 over GigE				
17	848	681	966	–
33	474	366	438	870
65	259	–	230	467
129	–	122	136	268
Single site (NCSA) run time (sec) using VMI1 over GM				
17	834	596	351	491
33	475	316	201	276
65	268	174	125	168
129	–	104	53	67
Cross site (NCSA + SDSC) run time (sec) using VMI1 over GigE				
17 (8 + 9)	835	700	1215	1937
33 (16 + 17)	478	369	421	890
65 (32 + 33)	261	217	390	667
129 (64 + 65)	–	131	183	428

results presented in Figure 5. This also highlights the fact that the serial overhead of the optimizer (RGA) does not significantly influence the overall performance trend. RGA-FEM performance results are shown in Table 2 for both single-site and cross-site cases using MPICH-VMI. The process count numbers in parentheses for the cross-site runs in the bottom rows of Table 2 show the number of processes used at each site. These results are obtained using two different builds of MPICH-VMI1 each using a different interface for intra-cluster messaging. The first is MPICH-VMI1 over GM and the second is over GigE. Some of the values in the table are missing due to system failure during these runs as the system was still in friendly user mode and not in full production mode. When the number of FEM simulation processes is greater than 1, we see that the runs using MPICH-VMI1 over GM (i.e., rows 5–8 in Table 2) perform better than MPICH-VMI1 over GigE (i.e., rows 1–4) for single-site runs since the Myrinet connection used by GM is faster than GigE. Again the runs that use four processes per FEM simulation show better performance with the faster GM interconnect, even as the total number of processes is increased. However, with the GigE interconnect the runs using two processes per FEM simulation perform better than four processes per FEM showing the influence of communication overhead ($\sim$ 20%) in FEM computations. This GigE communication overhead increases the overall run time by a factor of two when the number of processes per FEM is increased from 4 to 8. For the cross-site runs, the number of processes per FEM has a profound impact. For example, when we compare rows 1–4 from Table 2 with their

Table 3. Cross-site run times for the RGA-FEM simulations using single process for fine-grained FEM calculations.

Number of Processes	MPICH-G2	MPICH-G2	MPICH-VMI	MPICH-G2
	ANL	NCSA + SDSC		NCSA + SDSC + ANL
17	1571.90	2038.00	4781.16	2045.09
33	1147.966	1380.94	2432.95	1322.99
65	617.427	1021.18	1272.81	980.48
129	380.317	583.28	662.24	605.97

corresponding rows 9–12 (i.e., comparing single-site and cross-site use of GigE) we see that for the runs with one or two processes per FEM, the cross-site overhead is less than 8%. However, the overhead for the GigE interface is much higher (as high as 70%), particularly for runs with four or more processes per FEM. This is expected, as cross-site TCP/IP communication is slower.

In Table 3 we compare cross-site results obtained using both MPICH-G2 and MPICH-VMI2. These GA-EFM runs used a single process for the fine-grained FEM calculations. Different application parameters were used for these runs and hence a direct comparison could not be used between these results and those reported in Table 2. However, we do observe that the cross-site overhead is much higher for these runs when compared to the older runs (20–50% as opposed to 5–10%). This coincides with the ping-pong tests, which shows that the communication overheads are much higher since the TeraGrid went into production mode. The cross-site results show that MPICH-G2 is considerably faster than MPICH-VMI. Also, for cross-site runs involving three sites (NCSA, SDSC and UC/ANL) there is no additional overhead in going from two to three sites. This could be due to locality of the third site (UC/ANL) as it is physically in between the other two sites (closer to NCSA). The latency measured between SDSC and each of the other two sites is higher than the latency between NCSA and UC/ANL. We show only two cross-site results with MPICH-VMI2 as it was not available at ANL site at the time of these runs. The overhead of cross-site (two or three sites) runs increases as the number of processors is increased from 17 to 129 processors from $\sim$30% to $\sim$60% when compared with single site runs.

Additional results using multiple CPUs for fine-grained FEM calculations were deemed not to be useful in any case, as we want these fine-grained calculations local to a site. Also the results highlight the scalability and suitability of this application for grid computing.

6. Conclusions

We have examined the performance of a hierarchically parallel S/O (simulation-optimization) framework for solving groundwater inverse problems on the TeraGrid network involving three geographically distributed supercomputers. The S/O framework was built on top of grid-enabled MPI libraries. The results highlight the importance of having the fine-grained simulation operations confined to a single supercomputer site. The coarse-grained optimization component performs reasonably well across the TeraGrid site. Cross-site runs incurred small overhead (5–10%) compared to single-site runs if the fine-grained parallelism was confined to a single site and the network was not overstressed (early TG). Cross-site overhead could be as high as 50% during production phase of TG (current TG). In this paper we have used a problem that could be solved on a single machine so as to evaluate the potential overhead that could be incurred by going across geographically distributed machines. This might mislead one to question why someone would solve this problem on the TG when it could be solved more efficiently at a single site. There are many groundwater inverse problems that require GA population sizes in the thousands (e.g., source identification problems involving non-uniqueness, problems involving noise, etc.). These problems, while they could be solved on a single machine, will require thousands of processors in order to be completed in a timely manner. The typical queue wait time for requesting this number of processors at a single site could easily exceed the 5–50% overhead one would experience by requesting a smaller number of processors at each site, provided co-scheduling is available. Therefore we conclude that while one should exercise caution when selecting problems for distributing across sites, there will always be situations where this might reduce the overall computation time even with the added overhead.

The cross-site overhead could be reduced by the following enhancements that are planned in the future: (i) “chunking” the decision variables to reduce the latency overhead across sites, (ii) using the MPI-2 topology discovery mechanisms to create the level-1 groups and (iii) using MPI-2 connect/accept calls to separate the optimization module from the simulation module.

7. Acknowledgements

This work was partially supported by NSF under grants BES-0238623 and OCI-0222983. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors thank the TeraGrid for providing the computing resources.

8. References

[1] Abramson, D., Foster, I., Giddy, J., Lewis, A., Sosic, R., Sutherst, R., and White, N. 1997. The Nimrod computational workbench: A case study in desktop metacomputing. In *Proceedings of Australian Computer Science Conference* (*ACSC 97*), Macquarie University, Sydney, Australia, 1997.

[2] Allen, G., Dramlitsch, T., Foster, I., Karonis, N., Ripeanu, M., Seidel, E., and Toonen, B. 2001. Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus. *Proceedings of SC2001 Conference*, Denver, CO, November 10–16, 2001.

[3] Bangerth, W., Klie, H., Matossian, V., Parashar, M., and Wheeler, M. 2005. An autonomic reservoir framework for the stochastic optimization of well placement. *Cluster Computing* 8:255–269.

[4] Bear, J. 1972. *Dynamics of Fluids in Porous Media*. Environmental Science Series. New York: American Elsevier.

[5] Benger W., Foster, I., Novotny, J., Seidel, E., Shalf, J., Smith, W., and Walker, P. 1999. Numerical relativity in a distributed environment. In *Proceedings of Ninth SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[6] Boghosian B., Coveney, P., Dong, S., Finn, L., Jha, S., Karniadakis, G., and Karonis, N. 2006. Nektar, SPICE, and Vortonics: Using federated grid for large scale scientific applications". *Challenges of Large Applications in Distributed Environments* (*CLADE*) 2006:34–43.

[7] Casanova H, and Dongarra, J. 1997. Netsolve: A network-enabled server for solving computational science problems. 1997. *International Journal of Supercomputer Applications and High Performance Computing* 11(3):212–223.

[8] *Condor-G:* http://www.cs.wisc.edu/condor/condorg/

[9] *DARPA's Agent grid:* http://www.objs.com/agility/index.html

[10] Dong S, Karniadakis G, and Karonis, N. 2005. Cross-site computations on the Teragrid. *Computing in Science & Engineering* (*CiSE*) *Magazine* 7(5) Sept/Oct 2005:14–23.

[11] Dong, S., Insley, J., Karonis, N., Papka, M., Binns, J., and Karniadakis, G. 2006. Simulating and visualizing the human arterial system on the TeraGrid. *Future Generation of Computer Systems* (*FGCS*) 22(8):1011–1017.

[12] Dong, S., Karonis, N., and Karniadakis, G. 2006. Grid solutions for biological and physical cross-site simulations on the TeraGrid. *IEEE International Parallel & Distributed Processing Symposium* (*IPDPS*), 2006.

[13] *European Commission Enabling Grids for E-sciencE*: http://public.eu-egee.org/

[14] Foster I., and Kesselman, C. 1999. *The Grid, Blueprint for a New Computing Infrastructure*. San Francisco: Morgan Kauffman Publishers.

[15] Foster, I., and Kesselman, C. 1997. Globus. A metacomputing infrastructure toolkit. *International Journal of Supercomputing Applications* 11(2):115–128.

[16] *GLOBUS*: http://www.globus.org/

[17] Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

[18] Grama, A., Gupta A., Karypis G., and Kumar V. 2003. *Introduction to Parallel Computing*. 2^{nd} edition. Addison Wesley.

[19] Grimshaw, A. S., and Wulf, W. A. 1997. The Legion vision of a world wide virtual computer. *Communications of the ACM* 40(1):39–45.

[20] Gropp, W., Lusk W., and Skjellum, A. 1999. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. Cambridge, MA: The MIT Press.

[21] Huyakorn, P. S., and Pinder, G. F. 1983. *Computational Methods in Subsurface Flow*. Academic Press.

[22] Insley, J., Papka, M., Dong, S., Karniadakis, G., and Karonis, N. 2007. Runtime visualization of the human arterial tree. *IEEE Transactions of Visualization and Computer Graphics* (to appear).

[23] *Japan National Research Grid Initiative [NAREGI]*: http://www.naregi.org/index_e.html

[24] Feŕnandeza, J.-J., Garćıaa, I., Carazob, J.-M., and Marabini, R. 2007. Electron tomography of complex biological specimens on the grid. *Future Generation Computer Systems* 23:435–446.

[25] Karonis, N., Toonen, B., and Foster, I. 2003. MPICH-G2: A grid enabled implementation of the message passing interface, *Journal of Parallel and Distributed Computing* (*JPDC*) 63(5):551–563.

[26] Karonis, N., Papka, M., Binns, J., Bresnahan, J., Insley, J., Jones, D., and Link, J. 2003. High-resolution remote rendering of large datasets in a collaborative environment. *Future Generation of Computer Systems* (*FGCS*) 19(6):909–917.

[27] Kere, P., and Lento, J. 2005. Design optimization of laminated composite structures using distributed grid resources. *Composite Structures* 71:435–438.

[28] Lagana, D., Legato, P., Pisacane, O., and Vocaturo, F. 2006. Solving simulation optimization problems on grid computing systems. *Parallel Computing* 32:688–700.

[29] Luna, F., Nebro, A., and Alba, E. 2006. Observations in using grid-enabled technologies for solving multi-objective optimization problems. *Parallel Computing* 32:377–393.

[30] Mahinthakumar, G., and Saied, F. 2002. A hybrid MPI-OpenMP implementation of an implicit finite-element code on parallel architectures. *International Journal of High Performance Computing Applications* 16(4) Winter: 371–393.

[31] Mahinthakumar, G., and Saied, F. 1999. Implementation and performance analysis of a parallel multicomponent groundwater transport code. *Proceedings of the SIAM Parallel Processing Meeting*, San Antonio, TX, March 1999.

[32] Mahinthakumar, G., Hoffman F. M., Hargrove, W. W., Karonis, N. 1999. Multivariate geographic clustering in a metacomputing environment using Globus. *Proceedings of SC99 Conference on High Performance Networking and Computing*, Portland, Oregon, Nov 12–18, 1999.

[33] Mahinthakumar, G. 1999. *PGREM3D: Massively Parallel Codes for Groundwater Flow and Transport, Online User's guide*: http://www4.ncsu.edu/∼gmkumar/pgrem3d.pdf

[34] Mahinthakumar, G., Saied, F., and Valocchi, A. J. 1997. Comparison of some parallel Krylov solvers for large scale contaminant transport simulations. *High Performance Computing.* Society for Computer Simulation International, 1997:134–139.

[35] Melab, N., Cahon, S., and Talbi, E.-G. 2006. Grid computing for parallel bioinspired algorithms. *Journal of Parallel and Distributed Computing* 66(8):1052–1061.

[36] Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. 3^{rd} edition. New York: Springer-Verlag.

[37] Morrison, R. D. 2000. Application of forensic techniques for age dating and source identification in environmental litigation. *Environmental Forensics* 1:131–153.

[38] *MPICH-G2*: http://www.niu.edu/mpi/

[39] *MPICH-VMI*: http://vmi.ncsa.uiuc.edu/

[40] *NASA's Information Power Grid*: http://www.nas.nasa.gov/About/IPG/ipg.html

[41] Pakin, S., and Pant, A. 2000. VMI 2.0: A dynamically reconfigurable messaging layer for availability, usability, and management. *8th International Symposium on High-Performance Computer Architecture*, 2000.

[42] Parashar, M., Muralidhar, R., Lee, W., Arnold, D., Dongarra, J., and Wheeler, M. 2005. Enabling interactive and collaborative oil reservoir simulations on the grid. *Concurrency and Computation: Practice and Experience* 17:1387–1414.

[43] Poeter, E.P., and Hill, M. C. 1997. Inverse models: A necessary next step in groundwater modeling. *Groundwater* 35(2) Mar–Apr 1997:250–260.

[44] Saied, F., and Mahinthakumar, G. 1998. Efficient parallel multigrid based solvers for large scale groundwater flow simulations. *Computers Mathematics and Applications* 35(7):45–54.
[45] Sayeed, M., and Mahinthakumar, G. 2003. Hybrid optimization approaches for solving groundwater inverse problems in a parallel computing environment. *Proceedings of ASCE World Water Congress,* Philadelphia, PA, June 22–26.
[46] Schreiber, A., Metsch, T., and Kersken, H. 2005. A problem solving environment for multidisciplinary coupled simulations in computational grids. *Future Generation Computer Systems* 21(6):942–952.
[47] Sudholt, W., Baldridgea, K., Abramson, D., Enticott, C., Garicc, S., Kondricb, C., and Nguyenb, D. 2005. Application of grid computing to parameter sweeps and optimizations in molecular modeling. *Future Generation Computer Systems* 21(1):27–35.
[48] Sun, N.-Z. 1994. *Inverse Problems in Groundwater Modeling, Theory and Applications of Transport in Porous Media*. Kluwer Academic Publishers, 6,337 pp.
[49] *TeraGrid Network Performance*: https://network.teragrid.org/tgperf
[50] *tri-lab DISCOM grid*: http://www.cs.sandia.gov/discom/
[51] *US National Science Foundation TeraGrid* [TG]: http://www.teragrid.org
[52] *US Department of Energy Earth System Grid* [ESG]: http://www.earthsystemgrid.org
[53] *US Department of Energy Open Science Grid* [OSG]: http://www.opensciencegrid.org
[54] Yang, X., Hayes, M., Jenkins, K., and Cant, S. 2005. The Cambridge CFD grid for large-scale distributed CFD applications. *Future Generation Computer Systems* 21:45–51.

Mohamed Sayeed *is a computational researcher at Purdue University. He received his Ph.D. in Civil Engineering from North Carolina State University in 2004. His research interests are in high performance computing, numerical methods and inverse modeling.*

G. (Kumar) Mahinthakumar *is an associate professor of civil engineering at North Carolina State University. He obtained his Ph.D. in civil engineering from University of Illinois at Urbana-Champaign in 1995. His research interests are in high performance computing, inverse modeling and optimization and environmental fluid mechanics.*

Nicholas T. Karonis *received a B.S. in finance and a B.S. in computer science from Illinois University in 1985, an M.S. in computer science from Northern Illinois University in 1987 and a Ph.D. in computer science from Syracuse University in 1992. He is a professor at Northern Illinois University's computer science department and at Argonne's Mathematics and Computer Science Division as a resident associate guest working on the Globus Toolkit and MPICH-G2 and its successor MPIg. His current research interest is message-passing systems on computational grids.*