ELSEVIER

Advances in Water Resources 25 (2002) 243–261

Advances in Water Resources

www.elsevier.com/locate/advwatres

An efficient parallel-computing method for modeling nonisothermal multiphase flow and multicomponent transport in porous and fractured media

Yu-Shu Wu *, Keni Zhang, Chris Ding, K. Pruess, E. Elmroth [1], G.S. Bodvarsson

Earth Sciences Division, Lawrence Berkeley National Laboratory, MS 90-1116, Berkeley, CA 94720, USA

Received 23 May 2001; received in revised form 6 December 2001; accepted 8 January 2002

Abstract

This paper presents an efficient massively parallel scheme for modeling large-scale multiphase flow, multicomponent transport and heat transfer in porous and fractured reservoirs. The new scheme is implemented into a parallel version of the TOUGH2 code and the numerical performance is tested on a Cray T3E-900 (a distributed-memory parallel computer with 692 processors) and IBM RS/6000 SP (a distributed-memory parallel computer with 3328 processors). The efficiency and robustness of the parallel-computing algorithm are demonstrated by completing three over-million-cell simulations using site-specific data for site characterization. The first application is the development of a three-dimensional unsaturated zone numerical model simulating flow of moisture in the unsaturated zone of Yucca Mountain, Nevada; the second problem is the modeling of flow of moisture, gas, and heat at the same site. The third application is the study of flow-focusing phenomena through fractures for the same site. Simulation results show that the parallel-computing technique enhances modeling capability by several orders of magnitude in speedup of computing times for large-scale modeling studies. Published by Elsevier Science Ltd.

Keywords: Parallel computing; Groundwater modeling; Parallel reservoir simulation; Multiphase flow; Transport modeling

1. Introduction

Even with the significant advances made in both computational algorithms and computer hardware in reservoir modeling studies over the past half century, large-scale simulation of multiphase fluid and heat flow in heterogeneous reservoirs remains a challenge. The problem commonly arises from intensive computational efforts required for solving large sparse-matrix systems of highly nonlinear, discrete equations, resulting from detailed modeling investigations of reservoirs. In a continual effort to improve performance of reservoir simulators, massively parallel computing techniques have been developed that show promise in overcoming the limitations (such as constraints on problem size, and CPU time and space resolution) of reservoir simulation using reservoir simulators running on single-processor computers. High-performance, parallel-computing techniques have gradually received more attention in reservoir simulation and groundwater modeling communities, and a great deal of research effort has been devoted to this area in the past few decades.

Motivated by the desire to perform large-scale reservoir simulations using shared memory and distributed memory machines, researches on parallel reservoir simulation took place in the early 1980s. Those earlier researches, primarily in petroleum engineering (e.g., [6,25,27,44]), were focused on improving modeling capabilities. They used the then supercomputer's capability of vectorization with very different approaches to enhance modeling capabilities for dealing with large reservoir problems, as summarized by Coats [9].

More serious attempts to improve parallel-computing schemes and their applications were not made until the late 1980s and early 1990s. During this period of development, significant progress was made in several areas, including improvement in algorithms for parallelization [3,31,41,42,50], implementation into simulators using different computers [8,24,54], and applications to large-scale reservoir problems [45]. The parallel-computing techniques were implemented into

* Corresponding author. Fax: +1-510-486-5686.
E-mail address: yswu@lbl.gov (Y.-S. Wu).
[1] Now at Department of Computing Science and High Performance Computing Center North, Umeå University, Umeå, Sweden.

0309-1708/02/$ - see front matter. Published by Elsevier Science Ltd.
PII: S0309-1708(02)00006-4

black-oil type simulators with the IMPES solution scheme and compositional models (e.g., [5]). The largest problem (containing more than one million gridblocks) was demonstrated [59]. In the same time period, work on improving numerical algorithms associated with parallel computing was also carried out [4,28].

Up to the mid-1990s, progress in parallel computing was continuous. Several more sophisticated and efficient parallel reservoir simulators were developed [19–21]; and so were techniques for linear equation solution [26]. By the late 1990s, the parallel-computing reservoir-simulation technology was further improved and became more mature [12,53]. No longer relying solely on the mainframe multi-CPU or vector supercomputers, as in the early stage, distributed-memory parallel simulations were successfully implemented into and performed by Workstation clusters [23] and PC clusters [51]. Realistic field applications of parallel simulation techniques were demonstrated with multimillion gridblocks or megacell reservoir simulation [49].

In recent years, the demand on modeling capability in the fields of groundwater analysis, subsurface contamination study, environmental assessment, and geothermal engineering investigation has increased rapidly with the increase in computational intensity, as required by efforts in detailed site characterization and reliable performance assessment. Modeling capabilities based on the traditional single-CPU simulators have reached their limits with regard to what can be accomplished on those platforms. The high-performance computing technology has been increasingly viewed as an important, alternative modeling approach to resolving large-scale simulation problems [34]. Several parallel-computing methods have been developed and applied in these fields, including high-performance simulation of groundwater flow and contamination transport [1,29,35,46,58,62], multiphase flow modeling [15,61], algorithm development [2], and geothermal engineering application [60].

Despite significant progress in developing high-performance modeling tools over the past few decades, parallel-computing techniques have not been as widely applied in reservoir simulation as the traditional, one-processor simulators. One of the reasons for this may be the rapid advance in computing hardware, such as PCs and workstations, during the same period. Many types of small-to-intermediate-size reservoir problems can be adequately handled using a PC or workstation, which is much more affordable and available than a supercomputer. Secondly, many earlier-developed parallel schemes are machine dependent and are too difficult to implement on different computers or architectures. Perhaps more importantly, the reluctance to use parallel computing stems from the lack in development of efficient parallel schemes that can meet the needs of reservoir simulation. The severely nonlinear nature of reservoir dynamics, created by multiphase, multicomponent, and heat flow through heterogeneous porous media, poses a serious challenge to parallel-computing methodology. This is because the domain-partitioning scheme used in parallel computing may introduce additional non-linearity, as shown in two example problems of this work, to the global discretized equation for describing a fully coupled physical system, while handling many extra cross-bound flux terms between partitioned grid domains. This additional perturbation to the equation system from parallelization may sometimes override the numerical performance benefit from parallel computing itself, when solving highly nonlinear problems using non-Newton iteration or a less than fully implicit scheme. This may explain why parallel-computing technology has found more general application in solving geophysical inversion problems than in reservoir simulation, because equations governing geophysical processes are generally more linear than those for reservoir flow and transport.

Recent development in reservoir simulation and groundwater modeling identifies several rapidly growing areas of interest. These are: (1) a scaleup to simulate an oil or gas reservoir with a detailed, multimillion-to-multibillion-cell geological model [11,30]; (2) history-match, inverse, and optimization modeling for a reservoir with a long, complex production history that is subject to various physical and chemical processes; (3) application of Monte Carlo or other geostatistical modeling approaches; and (4) site characterization and long-term prediction of flow and transport processes in highly heterogeneous fractured media [57]. All of these model applications pose strenuous demands on modeling capability and numerical performance of reservoir simulators. In addition, future modeling investigations will tend to be more focused on analysis of fluid flow, multicomponent transport and heat transfer using much-refined grids with detailed spatial and temporal resolution, and a comprehensive description of complex geological, physical, and chemical processes at actual field sites. Therefore, continuous improvement in high-performance modeling tools is currently needed in both research and application.

In an effort to improve the current parallel-computing technology, this paper describes a new massively parallel computing scheme for conducting large-scale multiphase and multicomponent reservoir simulation. The primary objective of the new development is to present a machine- or platform-independent parallel algorithm that can be easily implemented into a mainframe supercomputer, a multi-processor workstation, or a cluster of PCs or workstations. Secondly, this work is intended to overcome numerical problems with the efficiency and robustness characteristic of the developed parallel-computing technology for handling severely nonlinear problems of multiphase flow and heat transfer in porous media. These goals are achieved by integrating and

optimizing the following procedures: (1) efficient domain partitioning; (2) parallel Jacobian calculations; (3) parallel-solving linearized equation systems; (4) fast communication and data exchange between processors; and (5) efficient memory sharing among processors.

This work presents our continual effort towards improving parallel reservoir simulation technology [14]. We here further develop and enhance the capabilities of a parallel version of the TOUGH2 family of codes [38] and evaluate the software on large-scale real-world application problems on Cray T3E and IBM SP massively parallel (MPP) computers. TOUGH2 is a general-purpose numerical simulation program for modeling multidimensional, multiphase, multicomponent fluid and heat flows in porous and fractured media. The parallel version of the TOUGH2 code preserves all the capabilities and features of its original one-processor version, based on the integral finite difference (IFD) method [33] for spatial discretization using an unstructured grid. Time is handled fully implicitly, and the resulting discretized nonlinear system of IFD equations for mass and energy balances is solved using the Newton method with an iterative or direct sparse matrix solver.

This paper presents an implementation and application of the massively parallel version of the TOUGH2 code. Also discussed are the efforts in reducing memory constraints by individual processors and in optimization to solve extremely large simulation problems. Application of the proposed parallel scheme is demonstrated through modeling three unsaturated flow problems at the Yucca Mountain site, a potential repository site for a high-level nuclear waste. The first two examples use a large, mountain-scale 3-D model, consisting of more than 10^6 gridblocks and 4×10^6 connections (interfaces), to simulate isothermal unsaturated flow, and nonisothermal flow of water, air and heat in an unsaturated zone domain of about 43 km^2 in area and 500–700 m in thickness of highly heterogeneous and fractured tuffs. In addition, these two test problems are also used to evaluate the numerical performance of the parallel scheme.

The third example is a study of the phenomena associated with focusing flow and discrete flowpaths through fractures at a potential repository. The 3-D model domain $(50\ m \times 50\ m \times 150\ m)$ is discretized into two million gridblocks for detailed modeling analyses of fractures on the scale of average fracture spacing, which cannot be simulated using the single-processor code. Simulation results for the three applications indicate that the parallel-computing technique implemented in TOUGH2 code is very efficient for both computing speedup and memory usage. Furthermore, results obtained with the refined-grid models provide detailed predictions of the ambient flow conditions as well as fracture flow behavior at the site, predictions that would not be possible to obtain with the single-processor version of the code.

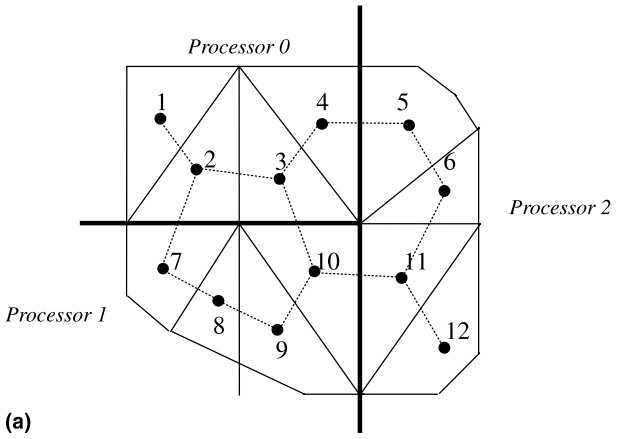
2. Methodology and implementation

The massively parallel computing technique of this work is based on a fully implicit formulation with Newton iteration and an ILU-based sparse iterative solver with BiCGSTAB/GMRES acceleration. This is because the fully implicit scheme has proven to be the most robust numerical approach in modeling multiphase flow and heat transfer in reservoirs over the past several decades. For a typical simulation with the fully implicit scheme and Newton iteration, such as within the TOUGH2 run, the most time-consuming steps of the execution consist of two parts that are repeated for each Newton iteration: (1) assembling the Jacobian matrix and (2) solving a linear system of equations. Consequently, one of the most important aims of a parallel code is to distribute computational time for these two parts. In addition, a parallel scheme must take into account grid node/element domain partitioning, grid node/element reordering, data input and output optimizing, and efficient message exchange between processors. These important parallel strategies and implementation procedures are discussed below.

2.1. Grid domain partitioning and gridblock reordering

Developing an efficient and effective method for partitioning unstructured grid domains is a first, critical step for a successful parallel scheme. In order to obtain optimal performance, the partitioning algorithm should ideally take the following five issues into account: (1) even computational load balance; (2) minimize the average (or total) communication volume; (3) even load balance in communication volume; (4) minimize the average (or total) number of neighboring processors; (5) even load balance in the number of neighboring processors. To find an optimal trade-off between these five issues, computer system characteristics, such as floating-point performance, and bandwidth and latency of the communication subsystem, must be taken into account. As this is a very complex problem, in general commonly used algorithms and software for partitioning large grids do not take all five issues into account. The current practice typically finds a trade-off between computation load balance and low total communication volume, even though they may not be theoretically optimal. More discussion on these issues and results for some state-of-the-art software is given in [13].

In a TOUGH2 simulation, a model domain is represented by a set of three-dimensional gridblocks (or elements), and the interfaces between any two gridblocks are represented by connections. The entire connection system of gridblocks is treated as an unstructured grid. From the connection information of a gridblock, an adjacency matrix can be constructed. The adjacency or connection structure of the model



(a)

Elements		1	2	3	4	5	6	7	8	9	10	11	12
xadj	1	2	5	8	10	12	14	16	18	20	23	26	27
adj		2	1,3,7	2,4,10	3,5	4,6	5,11	2,8	7,9	8,10	3,9,11	6,10,12	11

(b)

Fig. 1. An example of domain partitioning and CSR format for storing connections. (a) 12-elements domain partitioning on 3 processors, (b) CSR format.

meshes is stored in a compressed storage format (CSR). Fig. 1(a) shows the connection of a 12-element domain; Fig. 1(b) illustrates its corresponding CSR-format arrays.

We utilize three partitioning algorithms of the METIS package (version 4.0) [22] for our grid domain partitioning. The three algorithms are here denoted as the *K-way*, the *VK-way*, and the *Recursive* partitioning algorithm. *K-way* is used for partitioning a global mesh (graph) into a large number of partitions (more than 8). The objective of this algorithm is to minimize the number of edges that straddle different partitions. If a small number of partitions are desired, the *Recursive* partitioning method, a recursive bisection algorithm, should be used. *VK-way* is a modification of *K-way*, and its objective is to minimize the total communication volume. Both *K-way* and *VK-way* are multilevel partitioning algorithms.

Fig. 1(a) shows a scheme for partitioning a sample domain into three parts. Gridblocks are assigned to particular processors through partitioning methods and reordered by each processor to a local index ordering. Elements corresponding to these blocks are explicitly stored in the processor and are defined by a set of indices referred to as the processor's *update* set. The *update* set is further divided into two subsets: *internal* and *border*. Elements of the *internal* set are updated using only the information on the current processor. The *border* set consists of blocks with at least one edge to a block assigned to another processor. The *border* set includes blocks that would require values from other processors to be updated. The set of blocks not in the current processor, but needed to update components in the *border* set, is referred to as an *external* set. Table 1 shows the partitioning results, and one of the local numbering schemes for the sample problem presented in Fig. 1(a).

Table 1
Example of domain partitioning and local numbering

	Update		External
	Internal	Border	
Processor 0			
Gridblocks	1	2 3 4	5 7 10
Local numbering	1	2 3 4	5 6 7
Processor 1			
Gridblocks	8 9	7 10	2 3 11
Local numbering	1 2	3 4	5 6 7
Processor 2			
Gridblocks	6 12	5 11	4 10
Local numbering	1 2	3 4	5 6

Local numbering of gridblocks is carried out to facilitate the communication between processors. The numbering sequence is *internal* blocks, followed by *border* blocks, and finally by the *external* set. In addition, all *external* blocks from the same processor are in consecutive order.

Only nonzero entries of a submatrix for a partitioned mesh domain are stored on the processor. In particular, each processor stores only the rows that correspond to its *update* set. These rows form a submatrix whose entries correspond to variables of both the *update* set and the *external* set defined on this processor.

2.2. Organization of input and output data

The input data include hydrogeologic parameters and constitutive relations of porous media such as absolute and relative permeability, porosity, capillary pressure, thermophysical properties of fluids and rock, and initial and boundary conditions of the system. Other processing requirements include specification of space-discretized geometric information (grid) and various program options (such as computational and time-stepping parameters). In general, a large-scale, three-dimensional simulation requires at least several gigabytes of memory and the memory requirements should be distributed to all processors at input.

To make efficient use of the memory of each processor (considering that each processor has limited memory available), the input data files for the TOUGH2 simulation are organized in sequential format. There are two large groups of data blocks within a TOUGH2 mesh file, one with dimensions equal to the number of gridblocks and the other with dimensions equal to the number of connections (interfaces). Large data blocks are read one by one through a temporary full-sized array and then distributed to PEs (Processing Elements or processors). This method avoids storing all input data in a single PE (which may be too small) and greatly

enhances the I/O efficiency. Other small-volume data, such as simulation control parameters, are duplicated on all PEs.

Initialization parts of the parallel code require full-connection information such as for domain partitioning and local-connection index searching. These parts can exhaust the memory requirement for solving a large problem. Since the full-connection information is used only once at the beginning of a simulation, it may be better to handle these tasks in a preprocessing procedure.

2.3. Parallel computing strategy and program flowchart

In the TOUGH2 formulation, the discretization in space and time using the IFD leads to a set of strongly coupled nonlinear algebraic equations, which are solved by the Newton method. Within each Newton iteration, the Jacobian matrix is first constructed by numerical differentiation. The resulting system of linear equations is then solved using an iterative linear solver with different preconditioning procedures. The following is a brief discussion of our parallel algorithm for assembling and solving the linear systems of equations.

The fundamental goal of reservoir simulators is to solve spatially discretized, nonlinear equations of mass and energy, governing flow and transport processes in porous media. These discrete mass and energy balance equations solved by the TOUGH2 code can in general be written in residual form [37,38]:

$$R_n^\kappa(x^{t+1}) = M_n^\kappa(x^{t+1}) - M_n^\kappa(x^t) - \frac{\Delta t}{V_n}\left\{\sum_m A_{nm}F_{nm}^\kappa(x^{t+1}) + V_n q_n^{\kappa,t+1}\right\} = 0, \tag{1}$$

where the vector x^t consists of primary variables at time t, R_n^κ is the residual of component κ (heat is also regarded as a "component") for block n, M denotes mass or thermal energy per unit volume for a component, V_n is the volume of the block n, and q denotes sinks and sources of mass or energy. In addition, Δt denotes the current time step size, $t+1$ denotes the current time, A_{nm} is the interface area between blocks n and m, and F_{nm} is the flow between them. Eq. (1) is solved using the Newton method, leading to

$$-\sum_i \left.\frac{\partial R_n^{\kappa,t+1}}{\partial x_i}\right|_p \left(x_{i,p+1} - x_{i,p}\right) = R_n^{\kappa,t+1}(x_{i,p}), \tag{2}$$

where $x_{i,p}$ represents the value of ith primary variable at the pth iteration.

The Jacobian matrix as well as the right-hand side of (2) needs to be recalculated at each Newton iteration, and thus computational efforts may be extensive for a large simulation. In the parallel code, the assembly of linear equation system (2) is shared by all the processors, and each processor is responsible for computing the rows of the Jacobian matrix that correspond specifically to blocks in the processor's own *update* set. Computation of the elements in the Jacobian matrix is performed in two parts. The first part consists of computations relating to individual blocks (cumulative/source/sink terms). Such calculations are carried out using the information stored on the current processor and no communications with other processors are needed. The second part includes all computations relating to the connections or flow terms. Calculation of flow terms at elements in the *border* set needs information from the *external* set, which requires communication with neighboring processors. Before performing these computations, an exchange of relevant primary and secondary variables is required. For elements corresponding to *border* set blocks, one processor sends these elements to other related processors, which receive these elements as *external* blocks.

The Jacobian matrix for local gridblocks in each processor is stored in the distributed variable block row (DVBR) format, a generalization of the VBR format [7]. All matrix blocks are stored row-wise, with the diagonal blocks stored first in each block row. Scalar elements of gridblocks are stored in column major order. The data structure consists of a real vector and five integer vectors, forming the Jacobian matrix. Detailed explanation of the DVBR data format can be found in [47].

The linearized equation system arising at each Newton step is solved using an iterative linear solver from the Aztec package [47] with several different solvers and preconditioners to be selected. The work solving the global linearized equation is shared by all processors, with each processor responsible for solving its own portion of the partitioned domain equations.

During a simulation, the time steps are automatically adjusted (increased or reduced), depending on the convergence rate of iterations. In the parallel version code, the time-step size is calculated at the first processor (master processor, named PE0) after collecting necessary data from all processors. The convergence rates may be different in different processors. Only when all processors reach stopping criteria will the time march to the next time step. At the end of a time step or a simulation, the solutions obtained from all processors are then transferred to the master processor for output.

Communication of data between processors working on neighboring or connected gridblocks, partitioned into different domains, is an essential component of the parallel algorithm. Moreover, global communication is also required to compute norms of vectors, contributed by all processors, for checking the convergence. In addition to the communication that takes place inside the Aztec routine to solve the linear system, communication between neighboring processors is necessary to update

primary and secondary variables (for example, a new Jacobian matrix is calculated for each Newton iteration). A subroutine is used to manage data exchange between processors. When the subroutine is called by a processor, an exchange of vector elements corresponding to the *external* set of the gridblocks is performed. During time stepping or Newton iteration, exchange of external variables is also required for the vectors containing the primary and secondary variables. More discussion on the scheme used for data exchange is given in [14].

Fig. 2 shows an abbreviated overview of the program flowchart for the parallel version of TOUGH2, which is similar to the original version of the software [38]. Dynamic memory management, modules, array operations, matrix manipulation, and other FORTRAN 90 features are implemented on the parallel code. In particular, the message-passing interface (MPI) library [32] is used for message passing. Another important modification to the original code is in the time-step looping subroutine. This subroutine now provides general control of problem

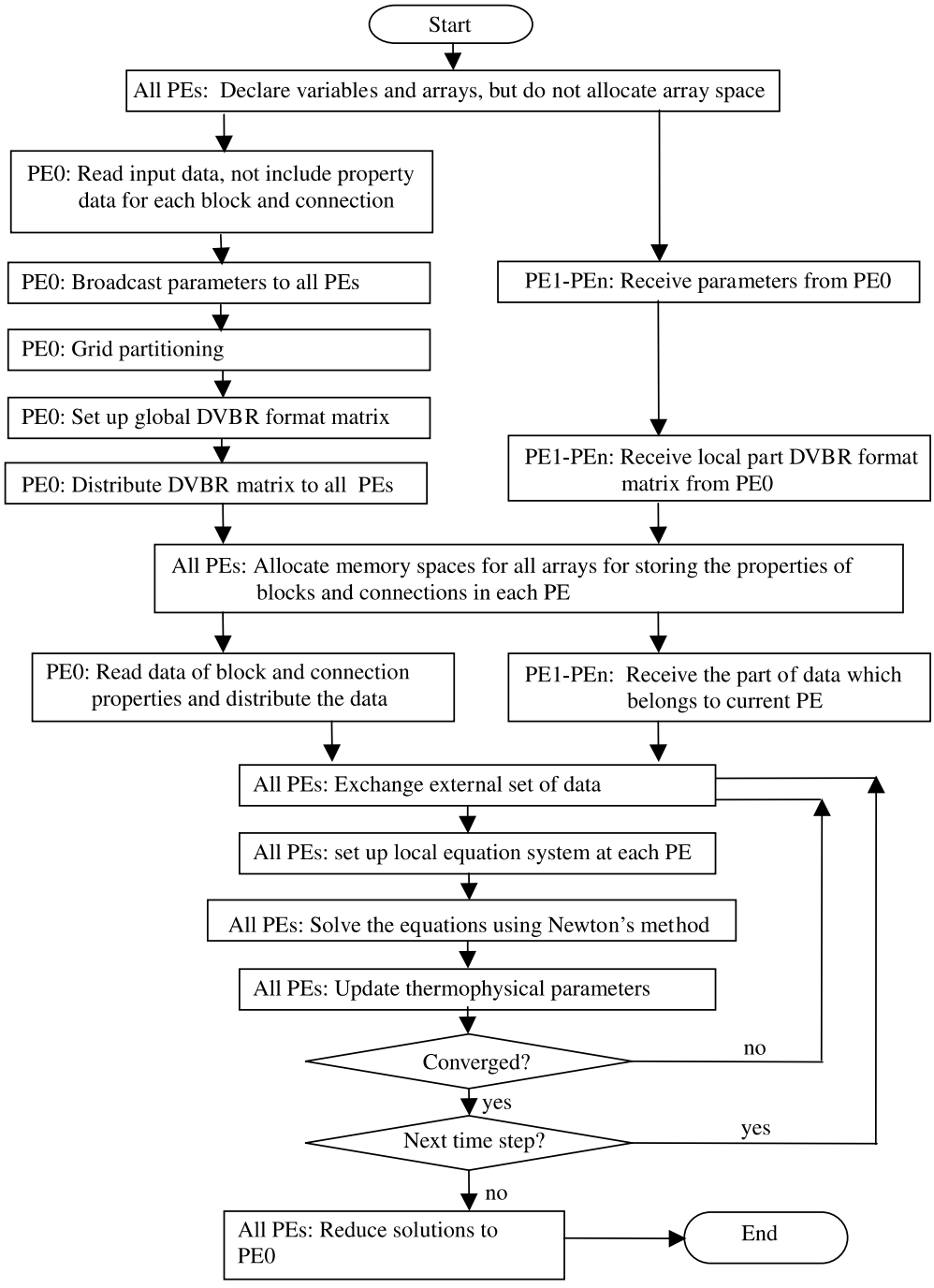


Fig. 2. Simplified flowchart of the parallel version TOUGH2.

initialization, grid partitioning, data distribution, memory requirement balancing among all processors, time stepping, and output options.

As shown in Fig. 2, all data input and output are carried out through the master processor, while the most time-consuming efforts (assembling the Jacobian matrix, updating thermophysical parameters, and solving the linear equation systems) are distributed to all processors. In addition, the memory requirements are also distributed to all processors. The effort of distributing both computing and memory requirements is essential for solving large-scale field problems.

2.4. Specification of initial and boundary conditions

Similar to a simulation using a one-processor code, the initial status of a multiphase system for the parallel version needs to be specified by initially assigning a complete set of primary variables to each gridblock. A commonly used procedure for specifying a capillary-gravity equilibrium condition is the restart option, in which a complete set of initial conditions is produced in a previous simulation with appropriate boundary conditions.

First-type or Dirichlet boundary conditions denote constant or time dependent phase pressure and saturation, mass fraction, and temperature conditions. These types of boundary conditions can be treated using the large-volume method, in which a constant pressure/saturation/concentration/temperature node is specified with a numerically large volume (while keeping all other geometric properties of the mesh unchanged). However, caution should be taken to: (1) identify phase conditions when specifying the "initial condition" for the large-volume boundary node and (2) distinguish upstream/injection from downstream/production nodes. Once specified, primary variables will be fixed at the large-volume boundary nodes, and the code handles these boundary nodes exactly like any other computational nodes. This physical-based treatment is easily handled by the parallel scheme as part of grid-domain partitioning initialization efforts.

Flux-type or Neuman boundary conditions are treated as constant or time-dependent sink/source terms, depending on the pumping (production) or injection condition, which can be directly added to Eq. (2). These terms are all implemented fully implicitly. This treatment of flux-type boundary conditions is especially useful when flux distribution along the boundary is known such as when dealing with surface infiltration. This method may also be used for an injection or pumping well connected to a single gridblock without injection or pumping pressures to be estimated. More general treatment of multilayered well conditions of production or injection for reservoir simulations can be handled using the virtual node method, as discussed in [55]. In this approach, a well connecting to many grid layers of production or injection is handled essentially by preprocessing mesh connection data, taking advantage of an unstructured TOUGH2 grid. This treatment is also parallelized with the grid-domain partitioning effort.

2.5. Handling fractured media

The TOUGH2 family of codes is equipped with a generalized technology for modeling flow and transport through fractured rock, based on the dual-continuum methodology [39,52]. This method treats fracture and rock-matrix flow and interactions, using a multi-continuum numerical approach, including the double- or multiporosity, the dual-permeability, and the more general "multiple interacting continua" (MINC) method.

The TOUGH2 model formulation and the parallel implementation are applicable (as discussed above) to both single-continuum and multi-continuum media. The TOUGH2 technology of handling flow in fractured reservoir by preprocessing a mesh to represent fracture and matrix domains [40] is directly applied to parallel simulation.

Once a proper mesh for the fracture-matrix system is generated, fracture and matrix blocks are specified to represent fracture or matrix domains, separately. Formally, they are treated exactly the same during grid partitioning or solution in the model (i.e., handled as a special case of the unstructured grid). However, physically consistent fracture and matrix properties, weighting schemes, and modeling conditions must be appropriately specified for fracture and matrix systems, respectively.

3. Application examples

We present three testing examples in this section to investigate computational performance and to demonstrate application of the parallel scheme. The three examples include:

(1) 3-D isothermal moisture flow in unsaturated fractured rock using more than one million gridblocks.
(2) 3-D gas, water and heat flow in unsaturated fractured rock using more than one million gridblocks.
(3) 3-D unsaturated liquid flow through fractures using Richards' equation and two million gridblocks.

3.1. 3-D site-scale example for unsaturated fluid flow

This large-scale 3-D model is used to evaluate the numerical performance of the parallel scheme and to demonstrate its application to modeling three-dimen-

sional flow within the unsaturated zone at Yucca Mountain, Nevada. The problem is based on the site-scale model developed for investigations of the unsaturated zone at Yucca Mountain, Nevada [56,57]. It concerns unsaturated flow through fractured rock under the ambient condition, using a 3-D, unstructured grid and a dual-permeability approach for handling fracture-matrix interactions. The unsaturated zone of Yucca Mountain has been investigated as a potential subsurface repository for storage of high-level radioactive wastes. The model domain of the unsaturated zone encompasses approximately 40 km^2 of the Yucca Mountain area is between 500 and 700 m thick, and overlies a relatively flat water table.

The 3-D model domain and the numerical grid used for this example are shown in plan view in Fig. 3. The irregular model grid uses a relatively fine gridblock size in the middle, repository, area and includes several nearly vertical faults. It has about 9900 blocks per grid layer representing both fracture and matrix continua, respectively, and about 60 computational grid layers in the vertical direction from land surface to water table. This results in a total of 1,075,522 gridblocks and 4,047,209 connections. A distributed-memory Cray T3E-900 computer equipped with 692 processors was used for this simulation example. Each processor of the computer has about 256 MB memory available and is capable of performing 900 million floating operations per second (Mflops/s).

The ground surface is regarded as the top model boundary, and the water table is regarded as the bottom boundary. Both top and bottom boundaries of the model are set as Dirichlet-type conditions. In addition, on the top boundary, a spatially varying water infiltration is applied to describe the net water recharge, with an average infiltration rate of 4.6 mm/yr over the model domain. The properties used for rock matrix and fractures for the dual permeability model, including two-

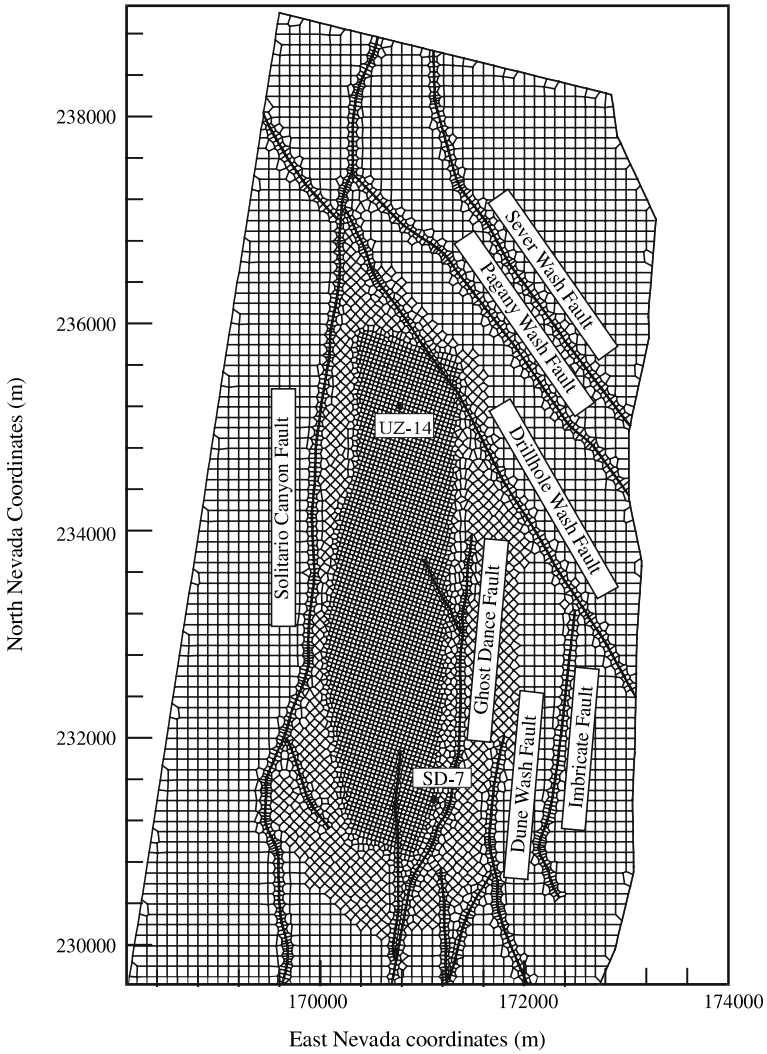


Fig. 3. Plan view of the 3-D site-scale model domain, gird and incorporated major faults.

phase flow parameters of fractures and matrix, were estimated from field tests and model calibration efforts, as summarized in [56].

Continual site characterization efforts show that fractured and matrix rocks at the site are extremely heterogeneous on any spatial scales from centimeter to kilometer. Flow characteristics within fracture and matrix systems are entirely different, because there are several-order-of-magnitude differences in permeability and water potentials between the two systems. Therefore, percolation through the two systems is also on very different time scales. In addition, moisture flow processes in the unsaturated zone are further complicated by existence of many high-permeability fault zones and low-permeability perched water areas. All of these, as demonstrated in previous modeling efforts, make solving the highly nonlinear equation for governing moisture flow at the site a computational challenge.

The linear system of equations arising at each Newton iteration is solved by the stabilized bi-conjugate gradient method. A domain decomposition-based preconditioner with ILUT [43] incomplete LU factorization was selected for preconditioning, and the *K-way* partitioning algorithm was used for partitioning the problem domain. The convergence criterion used for the iterative linear solver is

$$\frac{\|r\|_2}{\|b\|_2} \leqslant 10^{-6}, \tag{3}$$

where $\|\cdot\|_2 = \sqrt{\sum_{i=1}^{n} r_i^2}$, n is the total number of unknowns, and r and b are the residual and right-hand side, respectively.

The problem was designed to obtain the steady-state solution using the parallel code and to evaluate the code performance and improvement when using different numbers of processors. The simulation was conducted for 200 time steps. We ran the problem as a single-phase flow (water with Richards' equation using the EOS9 module of TOUGH2). The entire simulation was run on 64 processors for 3684 time steps to reach steady state, which was confirmed by examining the global mass balance.

The percolation flux through the repository horizon and below is one of the most important considerations in evaluating repository performance. Fig. 4 shows a comparison of the simulated flux distributions (Fig. 4(a)) along the repository, using the refined-grid model and parallel version of TOUGH2 (this work) and comparing its results to the coarse-grid results from 100,000 grid-blocks for the same geological domain (Fig. 4(b)) [56, nonparallel version]. In the figure, the dark blue color indicates higher values of percolation fluxes. The flux is defined in the figures as total mass flux through both

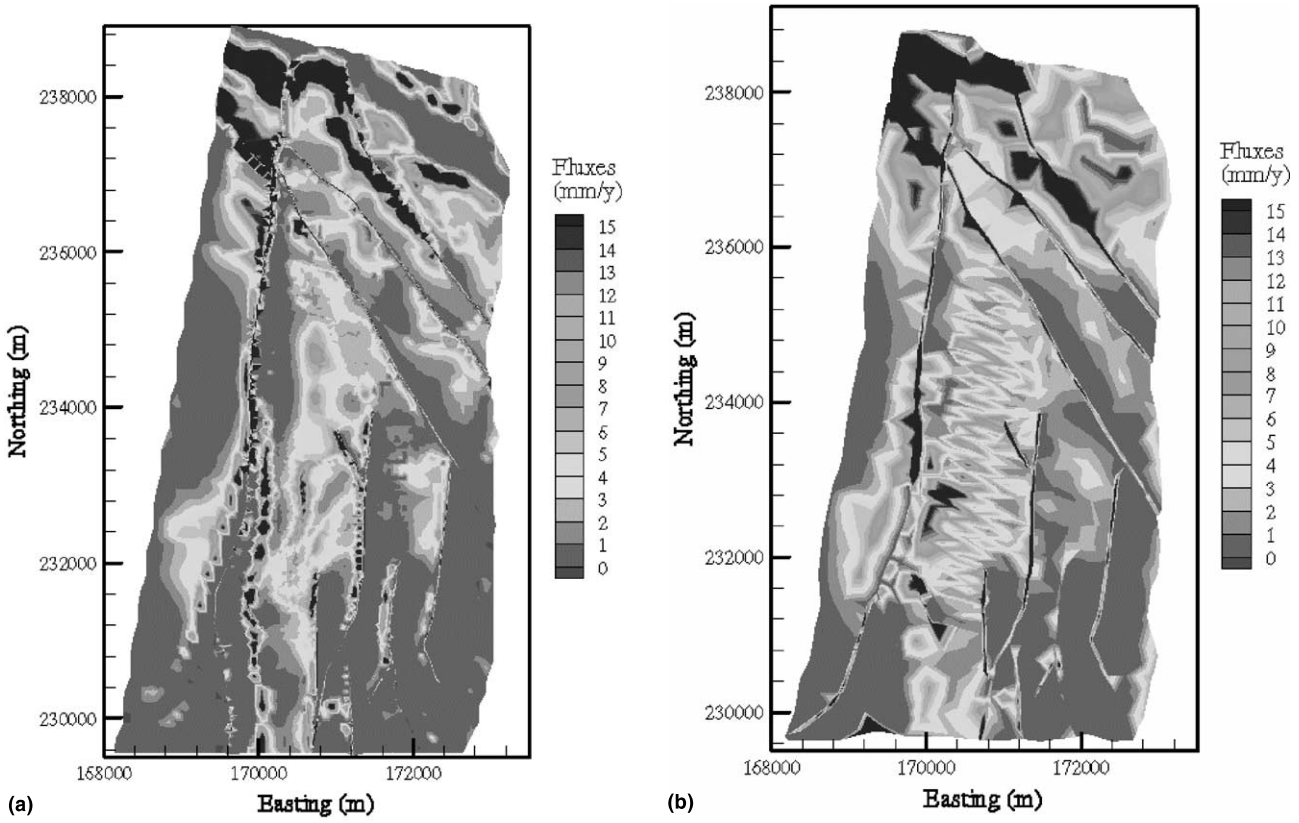


Fig. 4. Distribution of simulated vertical liquid fluxes at repository horizon: (a) using the refined-rid model (Fig. 3); (b) using the coarse-grid model [56].

fractures and matrix. Comparison of the simulation results in Figs. 5(a) and (b) indicates that the two models predict very different flow patterns in the repository area (along the middle of the model domain – see Fig. 3). The refined-grid model, with its much higher resolution, predicts much smaller percolation fluxes within the repository area. This indicates that more significant lateral flow occurs in the upper unit, above the repository horizon, when simulated by the refined grid model using the finer vertical grid spacing in these layers. These results will have a direct impact on performance assessment.

In performance evaluation, the unsaturated flow problem was solved using 32, 64, 128, 256, and 512 processors, respectively. Because of the automatic time-step adjustment, based on the convergence rate of the iteration process, the cumulative length of simulation times over 200 time-steps with different numbers of processors may be slightly different. However, the computational targets and efforts are similar, and comparing the performance of different numbers of processors with the same number of time steps is reasonable for evaluating the parallel-computing efficiency.

Table 2 shows the improvement (or reduction) in the total execution time as the number of processors increases. Simulations were run from 32 processors up to 512 processors by repeatedly doubling the number of processors. Clearly, the execution time is significantly reduced with the increase in processors. Table 2 also lists the times used for different computational tasks using different numbers of processors. When fewer than 128 processors are used, doubling the processor number reduces the total execution time by about a half. From the table, we can find that the best parallel performance is achieved in solving linear-equation systems. Note that data input and output of the program were carried out using a single processor; using a different number of processors might produce different overheads.

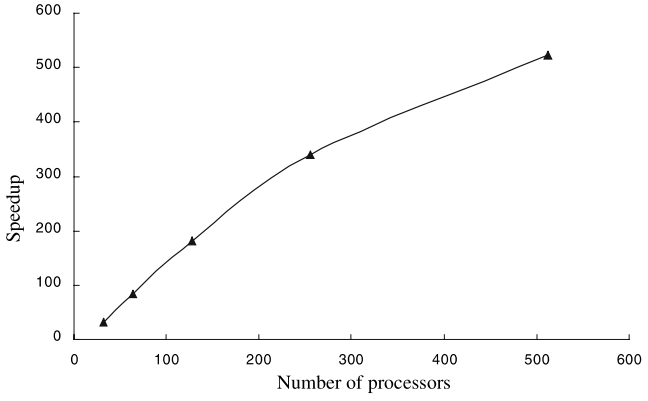


Fig. 5. Speedup for the 3-D site-scale model of isothermal unsaturated flow as function of number of processors used.

Fig. 5 illustrates the speedup of the parallel-code simulation versus processor numbers used. The speedup is defined to be relative to the performance of 32 processors as $32T_{32}/T_p$, where T_p denotes the total execution time using p processors. Speedup factors using 32–64, 128, 256, and 512 processors are 2.63, 2.16, 1.87, and 1.54, respectively. Better than linear speedup appears when the processor number doubles from 32 to 64, and to 128 with a speedup of 2.63 and 2.16. The overall speedup for 512 processors is 523. This behavior was extensively analyzed, including performance results for the different parts of the execution (hot spot analysis), for smaller problems in [14]. This follows from the following characteristics of the preconditioned linear solver. As the number of processors used increases, the size of a local domain for each processor decreases, which in turn has two effects. First, the effect of the preconditioner is decreased, so that the number of iterations in the linear solver increases. Second, the amount of work for computing the preconditioner is decreased as the ILUT factorization is only performed on the diagonal blocks of the Jacobian matrix (corresponding update elements). As the domains gets more and smaller, the ILUT factorization is computed on a more and more narrow block diagonal matrix, leaving more and more elements out of the computation. Most often, this effect is neglected as the reduced work for computing the preconditioner is often over-compensated by the increased amount of work by additional iterations in the linear solver. What is observed here (and in [14]) is that, up to a certain number of processors, the reduction in work from the second effect is large enough to influence the characteristics of the whole execution. As the number of processors grows, this effect is gradually reduced.

In comparison, the time requirement for the model-setup phase (input, distribution, and initialization) in Table 2 increases when the processor number is doubled

Table 2
Statistics of execution times (s) and iterations for the 3-D site-scale model problem of isothermal unsaturated flow for 200 time steps, using different numbers of processors

Number of processors	32	64	128	256	512
Input, distribution, and initialization	592.3	248.1	116.5	84.3	134.3
Update thermophysical parameters and setup Jacobian matrix	2659.2	1420.8	764.6	399.5	260.0
Solve linear equations	6756.7	2078.7	806.6	373.4	188.0
Total execution time	10,100.5	3844.3	1780.8	950.6	618.0
Total Newton iterations	415	415	424	424	423
Total Aztec iterations of solving linear equations	5059	5520	5735	6353	6281

from 256 to 512 (instead of generally decreasing). It indicates that a saturation point has been reached. These results from the increased communication overhead when increasing the number of processors such that we reach a point when the time saved from more processors in work sharing may be cancelled by communication overheads.

The partitioning algorithm may also significantly impact parallel code performance. It is generally desirable to distribute gridblocks evenly among the processors, with not only approximately the same number of internal gridblocks, but also roughly the same number of external blocks per processor. In our example, the domain-partitioning scheme implemented in the parallel code divided gridblocks almost evenly among processors. For example, on 128 processors, the average number of internal blocks is 8402 at each processor, with the maximum number 8657 and the minimum number 8156. It is only about a 6% difference between the maximum and minimum number. However, a close examination shows that considerable imbalance exists in the number of external blocks, which makes using unstructured grids difficult. The average number of external blocks is 2447, while the maximum number is as large as 3650 and the minimum as small as 918. This large range indicates that the communication volume can be four times higher for one processor than another. The imbalance in communication volume may lead to a considerable amount of time wasted on waiting for certain processors to complete their jobs during a solution. It was found [14] that the partitioning algorithms in the METIS package are generally effective in distributing the workload, but may result in certain imbalances between processors in terms of communication volume and number of messages.

3.2. 3-D site-scale example for two-phase fluid and heat flow

This second example is used to evaluate the numerical performance of the parallel scheme for highly nonlinear, two-phase fluid and heat flow in fractured rock. The model domain, geological model, numerical grid, and modeling for this problem are the same as the previous unsaturated flow case (Section 3.1). The difference is that the current problem models flows of water, gas and heat in a two active phase system using the multicomponent EOS3 module. The previous example handles only one active phase, using Richards's equation. Therefore, there are three equations per gridblock, and a total of $3 \times 1,075,522$ equations need to be solved per Newton iteration for the simulation. For this problem, another distributed-memory IBM SP computer equipped with 208 16–processor nodes (with a total of 3328 processors) was used. The 16 processors in each node of this computer share 16–64 GB memory, and each processor is capable of performing at 1.5 Gflops/s.

For practical application, the two-phase fluid and heat flow model can be used to estimate the natural hydrological and geothermal conditions at Yucca Mountain. This is an important part of modeling investigations, because liquid and gas (air and vapor) flow in the unsaturated zone of the mountain is affected by ambient temperature changes, geothermal gradients, and atmospheric and water table conditions. The boundary conditions for the nonisothermal flow model are treated similarly to those for the isothermal unsaturated flow case of Section 3.1 (i.e., top and bottom boundaries are treated as Dirichlet-type boundaries). Also the top boundary is subject to a spatially varying net water recharge. In addition, temperatures and gas pressures are needed (in this case) along these top and boundary surfaces. These constant (but spatially varying) temperatures on the top and bottom boundaries were determined from field observation [57]. On the other hand, pressure conditions at the bottom boundary of the water table were also calculated using observed gas pressure, whereas surface gas pressures are not fixed until a steady state is reached under given temperature, bottom pressure, and surface infiltration conditions.

In this problem, we selected the same options for the linear solver and the convergence tests as those used in Section 3.1. However, the calculation was designed to run to 1000 year simulation, using 16, 32, 64, 128, 256, 512 and 1024 processors, respectively.

Table 3 shows the statistics of parallel simulation performance with different numbers of processors. A comparison of the execution times (shown in Table 3), spent in simulations using different numbers of processors, indicates significant improvements achieved in numerical performance as processor numbers increase. Overall, the results of Table 3 are similar to those in Table 2: much more significant improvements are seen with processor numbers not too large ($\leqslant 256$) for this test problem. Again, the best numerical performance is in solving the linear systems as well as Jacobian matrix assembly.

Fig. 6 presents the speedup of the parallel-code simulation versus processor numbers used in the testing problem. Similarly, the speedup is defined to be relative to the performance with 16 processors (i.e., equal to 16 for using 16 processors). Fig. 6 also shows a better than linear speedup until the number of processors reaches 512. However, the speedup curve (as shown in Fig. 6) for more than 512 processors becomes flat, which is very different from Fig. 5. This deterioration in speedup performance using a larger number of processors results primarily from the increasing

Table 3
Statistics of execution times (s) and iterations for the 3-D site-scale model problem of nonisothermal two-phase fluid and heat flow for 1000 year simulation, using different numbers of processors

Number of processors	16	32	64	128	256	512	1024
Input, distribution, and initialization	1277.5	363.6	142.0	88.3	83.3	107.3	160.8
Update thermophysical parameters and setup Jacobian matrix	1328.8	1130.6	350.8	182.6	150.4	101.1	70.0
Solve linear equations	12,364.3	6413.1	1687.8	812.4	329.2	207.7	168.8
Total execution time	14,997.8	7960.4	2234.9	1137.6	618.8	472.3	461.7
Total Newton iterations	63	97	65	65	97	97	97
Total time steps	30	62	30	30	62	62	62
Total Aztec iterations/PE of solving linear equations	2022	2609	2211	3864	3017	4072	3226

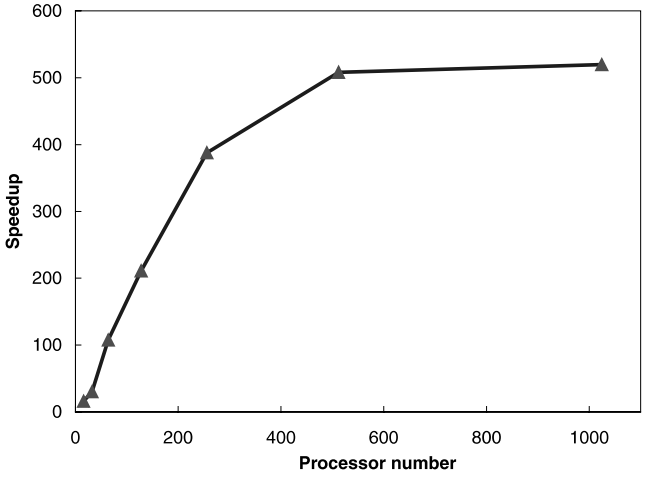


Fig. 6. Speedup for the 3-D site-scale model of nonisothermal two-phase fluid and heat flow as function of number of processors used.

communication in the linear solver. Since this performance test is performed for a relatively short simulated time, the one-time expense spent on the *input*, *distribution*, *and initialization* phase also contributes to this effect, as this part is relatively large for this problem. For example, the one-time expense takes 35% of the total execution time in the case of using 1024 processors, compared to 8.5% for 16 processors. Nevertheless, the effect of this one-time expense on the *input*, *distribution*, *and initialization* phase will diminish as total simulation times or time steps increase.

Examination of iteration numbers of Newton and linear solvers from Tables 2 and 3 indicates a general trend of both iteration numbers increasing as the number of processors used increases. For the linear solver, this effect is already explained by the reduced efficiency of the preconditioner as individual processors local domains become smaller. For the Newton iteration, an interpretation is that the problem may become more nonlinear, since all the simulations in comparisons were performed using the identical criteria for time-stepping and convergence. The increase in numbers required by Newton and linear solution iterations may be caused by additional stiffness introduced by more domain partitioning.

3.3. Example of 3-D liquid flow through unsaturated fractures

The third application is an analysis of flow focusing and discrete flow paths within the unsaturated zone of the Yucca Mountain site. Elevated levels of ^{36}Cl originating from atmospheric nuclear tests conducted in the 1950s and 1960s were found at several locations in an underground tunnel at Yucca Mountain [16], suggesting that preferential flow pathways may exist in the unsaturated rock of the mountain. Flow focusing along these preferential paths or well-connected fracture networks may play an important role in controlling patterns of percolation through highly fractured tuffs (such as in the Topopah Spring welded tuff [TSw] unit, which will house the repository drifts). Phenomena with flow focusing and discrete flowpaths in the TSw unit are thus considered to be of significant importance to potential repository performance [36].

Modeling of flow-focusing patterns is motivated by the needs of performance analysis, because water percolation in this unit cannot be readily measured at the site and has to be estimated using a model. On the other hand, grid resolution in the site-scale flow model of Wu et al. [56] or in the above example is generally too coarse to capture such localized flow phenomena. To quantify flow-focusing behavior, a stochastic fracture-continuum model is developed here to incorporate fracture data measured from the welded tuffs and to study flow allocation mechanisms and patterns. In particular, the model is used to assess the frequency and flux distributions of major water-bearing flow paths from the bottom of the Paintbrush nonwelded (PTn) unit, a unit immediately above the TSw.

The model domain of the 3-D flow problem, as shown in Fig. 7, is a 50 m $\times$ 50 m $\times$ 150 m parallelepiped, with the upper boundary at the bottom of the PTn and the lower boundary at the repository horizon. The dimension of the model was considered sufficient in the horizontal direction because the correlation length for variability in fracture permeability and spacing is approximately 1 m. The 150 m vertical extent of the model corresponds to an average distance from the interface between the PTn and TSw units to the repository

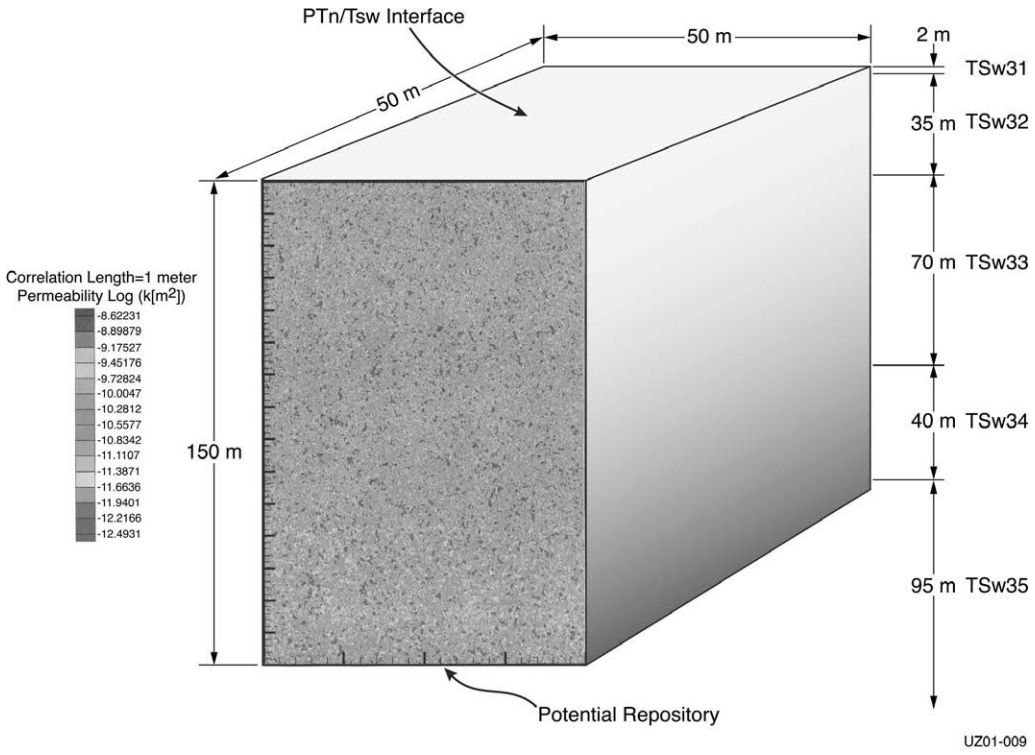


Fig. 7. Schematic of the 3-D model domain for fracture flow focusing studies, showing geological layers and fracture permeability distribution along a side surface.

horizon. The model covers five hydrogeological subunits from TSw31 to TSw35. The bottom of the PTn was chosen as the upper boundary because this unit behaves as a porous medium, leading to more uniform percolation flux to the units below. The four side boundaries are treated as no-flux boundaries, whereas the bottom boundary allows for gravitational drainage out of the model.

A refined grid is generated with each gridblock size of 0.5 m × 0.5 m × 0.75 m, in the same order in dimension as observed for fracture spacing. Such a refined grid captures flow behavior through individual discrete fractures. This leads to 2×10^6 gridblock elements and 6×10^6 connections with the 3-D grid.

In this study, several assumptions are made. The fracture network is modeled as a heterogeneous continuum, and flow through the matrix is neglected. This is because the matrix system in the unit has several-order-of-magnitude lower permeability and sensitivity studies indicate that the matrix has little impact under steady-state flow conditions in the rock unit. In addition, we focus our attention on investigating effects of heterogeneity fracture permeability only, because in prior modeling studies on seepage into drifts [18], heterogeneous-fracture-permeability fields are identified as a major factor impacting drift seepage, and large uncertainties exist with measured fracture permeability data.

Fracture permeability is prescribed stochastically for its spatial distribution, using measured air-permeability data. A spherical semivariogram model with empirical log-permeability semivariograms [10] is used to generate a three-dimensional, spatially distributed fracture-permeability multiplier. The permeability factor is correlated to each gridblock of 0.5 m × 0.5 m × 0.75 m, with a correlation length of 1 m. (Details on the methodology of generating stochastic fracture permeability at the site are discussed in [17].) The fracture permeability actually used is determined by multiplying the randomly generated multiplier to the fracture permeability (as a mean) of the gridblock's hydrogeological layer. The basic fracture properties, other than the spatially varying permeability multiplier, are assumed to be layer-wise constant and listed in Table 4 for the five geological layers (see Fig. 7).

In the table, k_f is fracture continuum permeability, α_f is van Genuchten's parameter [48] of capillary pressure for fractures, m_f is van Genuchten's parameter of fracture retention curves, ϕ_f is fracture porosity, and S_{rf} is residual liquid saturation in fracture.

The flow model results discussed below correspond to steady-state flow simulations carried out using the parallel TOUGH2 EOS9 module (Richards' equation). Different cases were run with the upper boundary prescribed with uniform infiltration flux of 1, 5, 25, and 100 mm/yr, respectively.

Table 4
Fracture properties used in the example problem

Model layer	k_f (m^2)	α_f (1/Pa)	m_f (–)	ϕ_f (–)	S_{rf} (–)
TSw31	3.21E – 11	2.49E – 4	0.566	5.5E – 3	0.01
TSw32	3.56E – 11	1.27E – 3	0.608	9.5E – 3	0.01
TSw33	3.86E – 11	1.46E – 3	0.608	6.6E – 3	0.01
TSw34	1.70E – 11	5.16E – 4	0.608	1.0E – 2	0.01
TSw35	4.51E – 11	7.39E – 4	0.611	1.2E – 2	0.01

Fig. 8 shows the distributions of simulated liquid fluxes extracted along a vertical slice from the 3-D model results, with the infiltration rate of 5 mm/yr specified on the top. The figure shows clearly that a large number of high-flux (dark blue), nearly vertical discrete flowing paths or *weeps* have developed along the vertical section. However, the simulated vertical flowpaths are generally short (less than 20 m) and show a discontinuous and sparse pattern. The lack in continuity throughout the vertical extent along the high-flux flowpaths is primarily caused by the use of a 2-D plot to display 3-D flow results. Actual weeps may flow freely into or from the third dimension, because of one additional freedom in dimension with a 3-D model. For comparison, Fig. 9 presents vertical flow results simulated using a 2-D model. The 2-D model, based on only one vertical cross-section of the 3-D model, uses the same parameter sets and boundary conditions as the 3-D model. Fig. 9 indicates that somehow vertically continuous major pathways (or weeps) appear in the upper layer (above the elevation of −110 m), but considerable changes are present in the lower layer. Comparison between the flow paths in Figs. 8 and 9 indicates that the 3-D modeling study is necessary for understanding the discrete-fracture behavior.

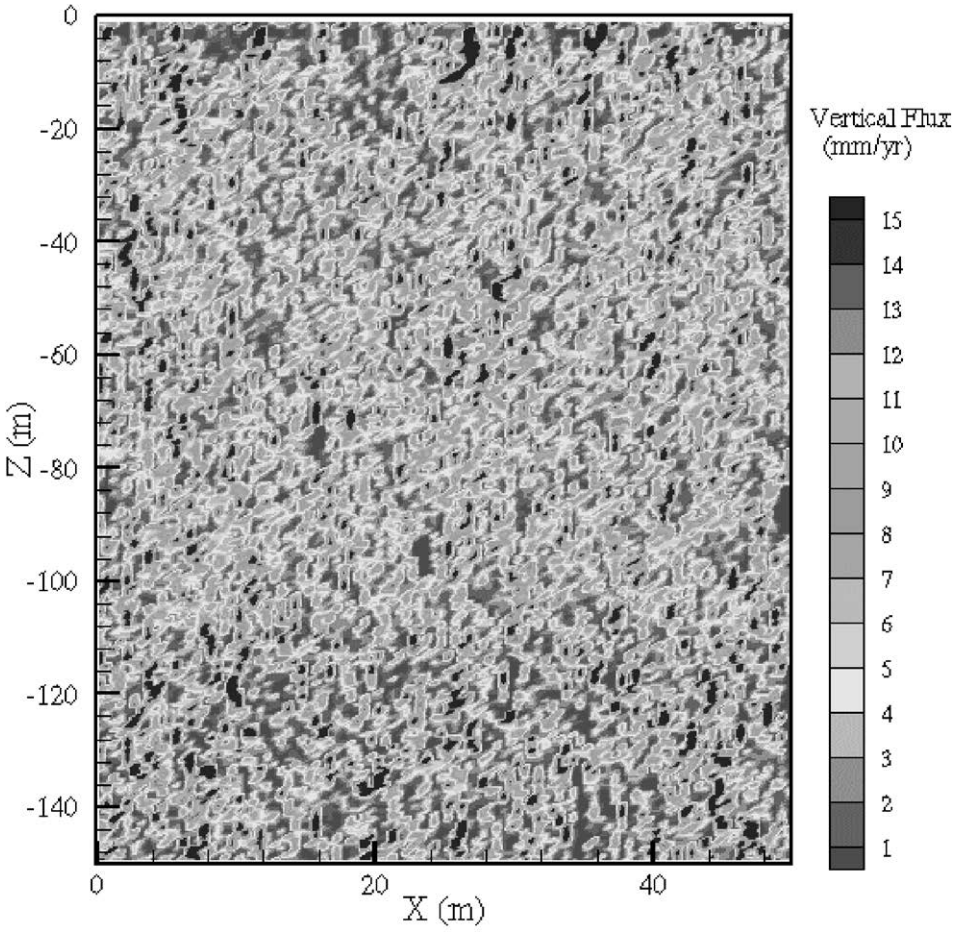


Fig. 8. Distribution of vertical liquid flux within the 2-D vertical cross-section of the 3-D model domain (with the infiltration rate of 5 mm/yr specified on the top boundary), indicating forming several high-flux, but short flow paths.

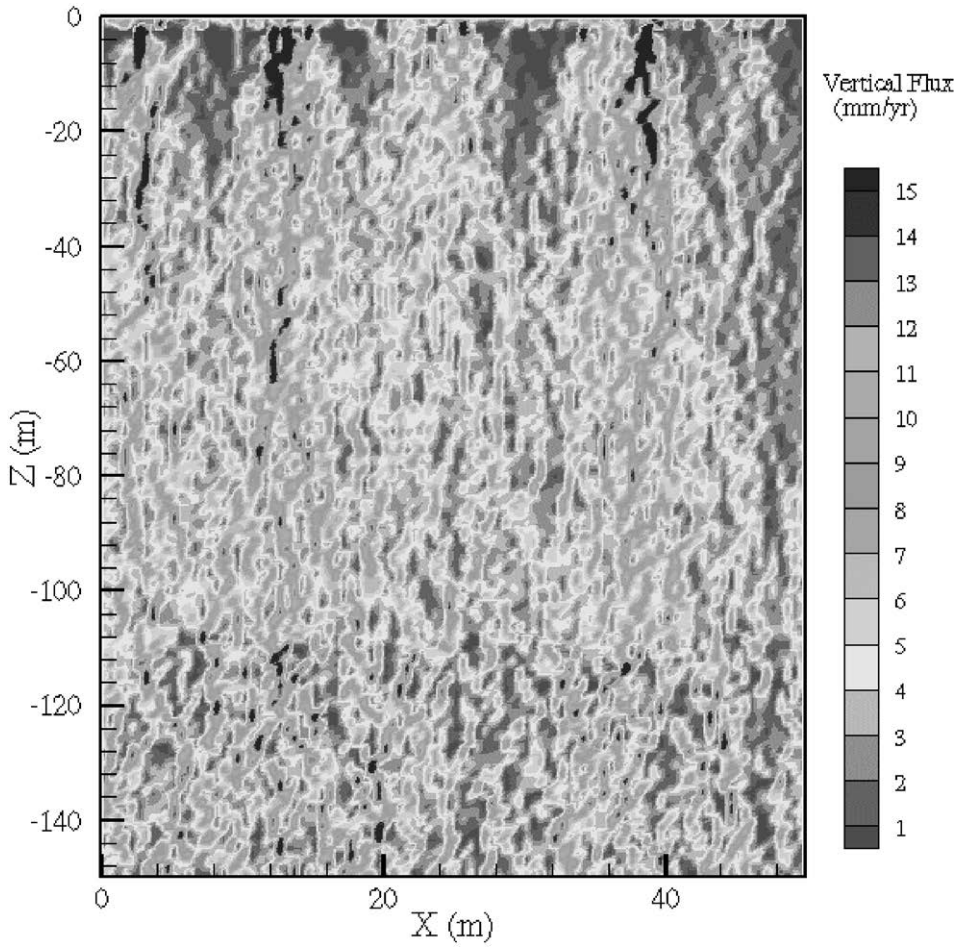


Fig. 9. Distribution of vertical liquid flux magnitude within the vertical cross-section using 2-D model results (with the infiltration rate of 5 mm/yr specified on the top boundary), indicating the forming of several high-flux and continuous flow paths.

Fig. 10 shows the horizontal distribution of simulated percolation fluxes along the bottom boundary of the model domain (i.e. on the repository level). The figure indicates a significant variability in both flux values and the sparseness of its distribution. The statistical analyses on weeps distributions at the repository horizon, shown in Fig. 10, will provide important input to the performance assessment of the repository.

Table 5 presents a summary of execution times consumed for two out of the four steady-state simulations on the Cray T3E-900 machine. The other two runs are not included in the table because they were interrupted by other jobs, preventing accurate CPU times from being recorded. The statistics in Table 5 show that it took about 2 h to complete a two-million-gridblock simulation, demonstrating the efficiency of the parallel implementation. In addition, Table 5 shows that CPU times used in calculations of secondary parameters and assembly of the Jacobian are longer than those used in solving linear equations for this problem. This is very different from what we observe using a single-processor simulator with the Newton iteration scheme. The same behavior can also be found from Table 2, as the number of processors increases. This is because of the increase in overheads caused by communication between processors during the set up of the linear equation system. Note that the longer execution times for the case 1 mm/yr infiltration (compared with 25 mm/yr in Table 5) result from lower infiltration leading to a drier unsaturated fracture or a more nonlinear problem. Therefore, a smaller infiltration case takes in general a longer simulation time for the example.

4. Summary and concluding remarks

This paper describes a new massively parallel scheme, along with its implementation into the TOUGH2 code and application examples to large-scale reservoir simulations. We present a portable parallel implementation in Fortran 77 and 90 using message passing interface (MPI) for inter-processor communication. As MPI is currently available for all major computing platforms, this ensures portability over a

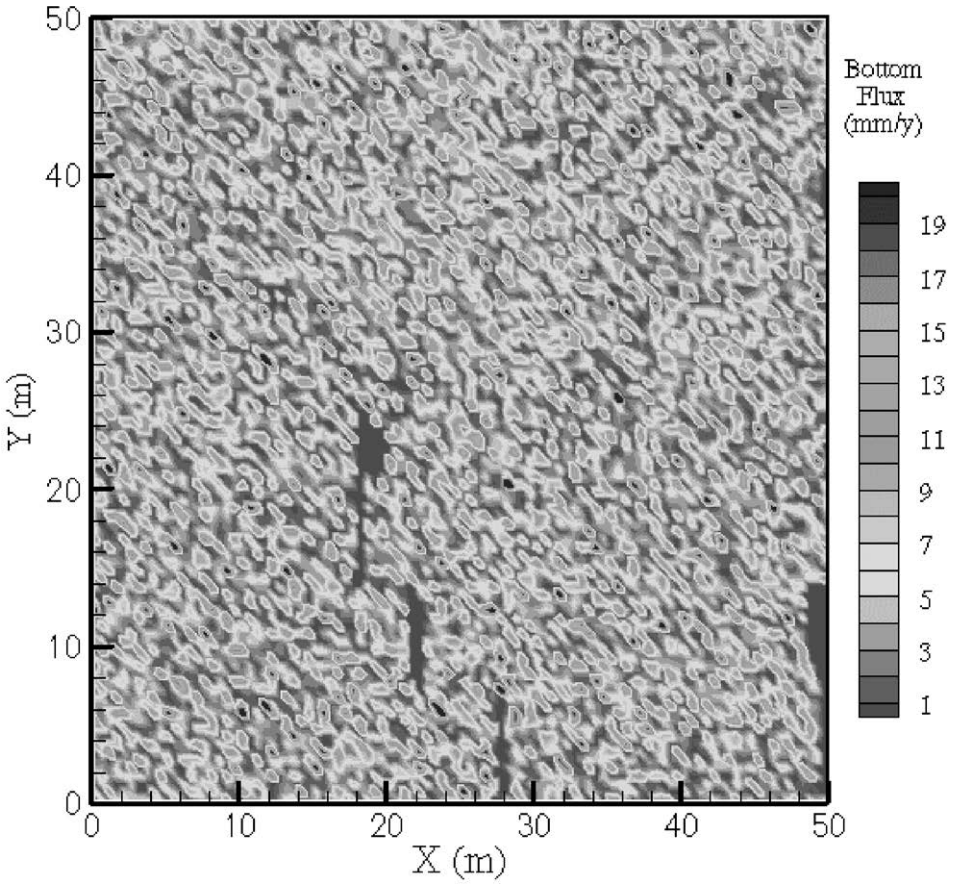


Fig. 10. Distribution of vertical liquid fluxes at the bottom boundary or the repository level (with the infiltration rate of 5 mm/yr specified on the top boundary), indicating possible weeps patterns.

Table 5
Summary of execution times (s) used for two simulations of the 3-D fracture flow focusing problem, using 128 processors

Simulation	1 mm/yr infiltration	25 mm/yr infiltration
Input, distribution, and initialization	218	218
Update parameters and setup Jacobian matrix	4865	4655
Solve linear equations	2109	1194
Total execution time	8111	6960

whole range of computers, including shared memory machines, clusters of PCs or workstations and more classical high performance computing systems. To further improve the parallel performance, we devote intensive effort to overcoming the efficiency problems with existing parallel-computing technology in handling highly nonlinear multiphase flow and heat transfer in porous media. In particular, in the current scheme, both computing and memory requirements are distributed among and shared by all processors of a multi-CPU computer or cluster. The best numerical performance has been achieved by integrating and optimizing the following procedures: (1) efficient domain partitioning; (2) parallel assembly of the Jacobian matrix; (3) parallel preconditioned iterative linear solver; (4) fast communication and data exchange between processors; and (5) efficient utilization of the processors aggregated memory.

In this study, the efficiency and improvement derived from parallel-computing technology are evaluated by three large-scale applications of flow in the unsaturated zone at Yucca Mountain. The first problem uses more than a million gridblocks to develop a mountain-scale moisture flow model. The second application models two-phase (liquid and gas) and heat flow with the same mountain-scale model. The third example is designed to investigate detailed fracture behavior within the one particular geological unit housing the repository drifts. Test results conclude that the massively parallel computing scheme of this work has achieved great im-

provement in computational efficiency and robustness for dealing with highly nonlinear problems for large-scale reservoir simulation problems and in particular has enhanced our modeling capability significantly. The parallel scheme implementation enables us to use much more refined spatial discretization and obtain many more insights into flow problems than coarse-grid models would. None of the problems presented here would be possible to solve without this parallel software.

Another intention of this work is to generate more interest among scientists and engineers in studying and applying parallel-computing techniques to solve real-world modeling problems. Rapid advances in computing-hardware technology make high-performance computers (including PCs and workstations) readily available for parallel-computing efforts. On the other hand, ever-increasing demands on reservoir simulator performance efficiency (needed in conducting detailed reservoir investigations) will lead to further development and wider application of the parallel-computing technology in the near future.

Acknowledgements

The authors would like to thank Jianchun Liu, Andre Unger and Dan Hawkes for their review of this paper. The authors are grateful to Lehua Pan for his help in designing the 3-D grid used for the first field problem. We would also like to thank the three anonymous reviewers and the Editorial Board member of *Advances in Water Resources Journal* for their insightful and constructive comments and suggestions for improving the manuscript. This work was supported by the Laboratory Directed Research and Development (LDRD) program of Lawrence Berkeley National Laboratory. The support is provided to Berkeley Lab through the US Department of Energy Contract No. DE-AC03-76SF00098.

References

[1] Ashby SF, Bosl WJ, Falgout RD, Smith SG, Tompson AFB. Numerical simulation of groundwater flow and contaminant transport on the CRAY T3D and C90 supercomputers. Int J High Performance Comput Appl 1999;13(1):80–93.

[2] Ashby SF, Falgout RD. Parallel multigrid preconditioned conjugate gradient algorithms for groundwater flow simulations. Nucl Sci Eng 1996;124(1):145–59.

[3] Barua J, Horne RN. Improving the performance of parallel (and series) reservoir simulation. In: Proceedings of Tenth SPE Symposium on Reservoir Simulation, Houston, TX, February 6–8, 1989. p. 7–18, Paper SPE 18408.

[4] Bhogeswara R, Killough JE. Parallel linear solvers for reservoir simulation: generic approach for existing and emerging computer architectures. In: Proceedings of 12th SPE Symposium on Reservoir Simulation, New Orleans, LA, 28 February–March 3, 1993. p. 71–82, Paper SPE 25240.

[5] Briens FJL, Wu CH, Gazdag J, Wang HH. Compositional reservoir simulation in parallel supercomputing environments. In: Proceedings of Eleventh SPE Symposium on Reservoir Simulation, Anaheim, CA, February 17–20, 1991. p. 125–33, Paper SPE 21214.

[6] Calahan DA. Vectorized direct solvers for 2-D grids. In: Proceedings of Sixth SPE Symposium on Reservoir Simulation, New Orleans, LA, February 1–3, 1982. p. 489–97, Paper SPE 10522.

[7] Carney S, Heroux M, Li G. A proposal for a sparse BLAS toolkit. Technical Report, Cray Research Inc., Eagen, MN, 1993.

[8] Chien MCH, Northrup EJ. Vectorization and parallel processing of local grid refinement and adaptive implicit schemes in a general purpose reservoir simulator. In: Proceedings of 12th SPE Symposium on Reservoir Simulation, New Orleans, LA, 28 February–March 3, 1993. p. 279–90, Paper SPE 25258.

[9] Coats KH. Reservoir simulation. In: Petroleum engineering handbook. Richardson, TX: Society of Petroleum Engineers; 1987. p. 48-1–48-20 [chapter 48].

[10] Deutsch CV, Journel AG. GSLIB geostatistical software library and user's guide. New York: Oxford University Press; 1992.

[11] Dogru AH. Megacell reservoir simulation. J Pet Technol 2000 May;52(5):54–60, Paper SPE 57907.

[12] Dogru AH, Li KG, Sunaidi HA, Habiballah WA, Fung L, Al-Zamil N, Shin D, McFonald AE, Srivastava NK. A massively parallel reservoir simulator for large scale reservoir simulation. In: Proceedings of Fifteenth SPE Symposium on Reservoir Simulation, Houston, TX, February 14–17, 1999. p. 73–92, Paper SPE 51886.

[13] Elmroth E. On grid partitioning for a high performance groundwater simulation software. In: Engquist, et al., editors. Simulation and visualization on the grid. Lecture Notes in Computational Science and Engineering, No. 13. Berlin: Springer; 2000. p. 221–33.

[14] Elmroth E, Ding C, Wu YS. High performance computations for large-scale simulations of subsurface multiphase fluid and heat flow. J Supercomputing 2001;18(3):233–56.

[15] Eppstein M, Dougherty DE. Comparative study of PVM workstation cluster implementation of a two-phase subsurface flow model. Adv Water Resour 1994;17(3):181–95.

[16] Fabryka-Martin JT, Dixon PR, Levy S, Liu B, Turin HJ, Wolfsberg AV. Systematic sampling for chlorine-36 in the exploratory studies facility. Report LA-CST-TIP-96-001, Milestone 3783AD, Los Alamos National Laboratory, Los Alamos, NM, 1996.

[17] Finsterle S. Using the continuum approach to model unsaturated flow in fractured rock. Water Resour Res 2000;36(8):2055–66.

[18] Finsterle S, Ahlers CF, Trautz RC. Seepage calibration model and seepage testing data. Report MDL-NBS-HS-000004 REV01. Lawrence Berkeley National Laboratory, Berkeley, CA; and Las Vegas, Nevada, CRWMS M&O, 2000.

[19] Ghori SG, Wang CH, Lim TM, Pope GA, Sepehrnoori K. Computational reservoir simulation on CM-5 and KSR-1 parallel machines. In: Proceedings of 13th SPE Symposium on Reservoir Simulation, San Antonio, TX, February 12–15, 1995. p. 479–89, Paper SPE 29140.

[20] Hemanth-Kumar K, Young LC. Parallel reservoir simulator computations. In: Proceedings of 13th SPE Symposium on Reservoir Simulation, San Antonio, TX, February 12–15, 1995. p. 101–10, Paper SPE 29104.

[21] Kaarstad TS, Froyen J, Bjorstad P, Espedal M. A massively parallel reservoir simulator. In: Proceedings of 13th SPE Symposium on Reservoir Simulation, San Antonio, TX, February 12–15, 1995. p. 467–78, Paper SPE 29139.

[22] Karypsis G, Kumar V. METIS. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, V4.0. Technical

Report, Department of Computer Science, University of Minnesota, 1998.
[23] Killough J, Commander D. Scalable parallel reservoir simulation on a windows NT™-based workstation cluster. In: Proceedings of Fifteenth SPE Symposium on Reservoir Simulation, Houston, TX, February 14–17, 1999. p. 41–50, Paper SPE 51883.
[24] Killough JE, Bhogeswara R. Simulation of compositional reservoir phenomena on a distributed memory parallel computer. In: Proceedings of Eleventh SPE Symposium on Reservoir Simulation, Anaheim, CA, February 17–20, 1991. p. 69–82, Paper SPE 21208.
[25] Killough JE, Levesque JM. Reservoir simulation and the in-house vector processor experience for the first year. In: Proceedings of Sixth SPE Symposium on Reservoir Simulation, New Orleans, LA, February 1–3, 1982. p. 481–7, Paper SPE 10521.
[26] Kohar G, Killough JE. An asynchronous parallel linear equation solution technique. In: Proceedings of 13th SPE Symposium on Reservoir Simulation, San Antonio, TX, February 12–15, 1995. p. 507–20, Paper SPE 29142.
[27] Levesque JM. Reservoir simulation on supercomputers. In: Proceedings of Sixth SPE Symposium on Reservoir Simulation, New Orleans, LA, February 1–3, 1982. p. 471–9, Paper SPE 10520.
[28] Li KMG. A linear equation solvers for massively parallel computers. In: Proceedings of 12th SPE Symposium on Reservoir Simulation, New Orleans, LA, 28 February–March 3, 1993. p. 83–8, Paper SPE 25241.
[29] Li L, Barry DA, Morris J, Stagnitti F. CXTANNEAL: an improved program for estimating solute transport parameters. Environ Modeling and Software 1999;14(1–3):607–11.
[30] Li D, Beckner B. Optimal uplayering for scaleup of million-cell geological models. J Pet Technol 2001;53(4):75–7.
[31] Meijerink JA, van Daalen DT, Hoogerbrugge PJ, Zeestraten RJA. Towards a more efficient parallel reservoir simulator. In: Proceedings of Eleventh SPE Symposium on Reservoir Simulation, Anaheim, CA, February 17–20, 1991. p. 107–16, Paper SPE 21212.
[32] Message Passing Interface Forum. MPI: A message-passing interface standard. Int J Supercomputing Appl High Performance Comput 1994;8(3–4).
[33] Narasimhan TN, Witherspoon PA. An integrated finite difference method for analyzing fluid flow in porous media. Water Resour Res 1976;12(1):57–64.
[34] Oduro P, Nguyen HV, Nieber JL. Parallel computing applications in groundwater flow: an overview, American society of agricultural engineers. In: Proceedings of the 1997 ASAE Annual International Meeting Part 1 (of 3), Minneapolis MN, August 10–14, 1997. p. 0145–66.
[35] Pini G, Putti M. Parallel finite element laplace transport method for the non-equilibrium groundwater transport equation. Int J Numer Meth Eng 1998;40(14):2653–64.
[36] Pruess K. A mechanistic model for water seepage through thick unsaturated zones of fractured rocks of low permeability. Water Resour Res 1999;35(4):1039–52.
[37] Pruess K, Oldenburg C, Moridis G. TOUGH2 user's guide, V2.0. Lawrence Berkeley National Laboratory Report LBNL-43134, Berkeley, CA, 1999.
[38] Pruess K. TOUGH2 – a general-purpose numerical simulator for multiphase fluid and heart flow. Lawrence Berkeley Laboratory Report LBNL-29400, Berkeley, CA, 1991.
[39] Pruess K, Narasimhan TN. A practical method for modeling fluid and heat flow in fractured porous media. Soc Pet Eng J 1985;25:14–26.
[40] Pruess K. GMINC – a mesh generator for flow simulations in fractured reservoirs. Report LBL-15227, Berkeley, Lawrence Berkeley National Laboratory, CA, 1983.
[41] Quandalle P, Moriano S. Vectorization and parallel processing of models with local refinement. In: Proceedings of Eleventh SPE Symposium on Reservoir Simulation, Anaheim, CA, February 17–20, 1991. p. 93–105, Paper SPE 21210.
[42] Rutledge JM, Jones DR, Chen WH, Chung EY. The use of a massively parallel SIMD computer for reservoir simulation. In: Proceedings of Eleventh SPE Symposium on Reservoir Simulation, Anaheim, CA, February 17–20, 1991. p. 117–24, Paper SPE 21213.
[43] Saad Y. ILUT: a dual threshold incomplete ILU factorization. Numer Linear Algebra Appl 1994;1(4):387–402.
[44] Scott AJ, Sutton BR, Dunn J, Minto PW, Thomas CL. An implementation of a fully implicit reservoir simulator on an icl distributed array processor. In: Proceedings of Sixth SPE Symposium on Reservoir Simulation, New Orleans, LA, February 1–3, 1982. p. 523–33, Paper SPE 10525.
[45] Scott SL, Wainwright RL, Raghavan R, Demuth H. Application of parallel (MIMD) computers to reservoir simulation. In: Proceedings of Ninth SPE Symposium on Reservoir Simulation, San Antonio, TX, February 1–4, 1987, Paper SPE 16020.
[46] Tsai WF, Shen CY, Fu HH, Kou CC. Study of parallel computation ground-water solute transport. J Hydrol Eng 1999;4(1):49–56.
[47] Tuminaro RS, Heroux M, Hutchinson SA, Shadid JN. Official Aztec user's guide, ver 2.1. Albuquerque, NM: Massively Parallel Computing Research Laboratory, Sandia National Laboratories; 1999.
[48] van Genuchten MTh. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. Soil Sci Soc Am J 1980;44(5):892–8.
[49] Vertiere S, Quettier L, Samier P, Thompson A. Application of a parallel simulator to industrial test cases. In: Proceedings of Fifteenth SPE Symposium on Reservoir Simulation, Houston, TX, February 14–17, 1999. p. 93–105, Paper SPE 51887.
[50] Wallis JR, Foster JA, Kendall RP. A new parallel iterative linear solution method for large-scale reservoir simulation. In: Proceedings of Eleventh SPE Symposium on Reservoir Simulation, Anaheim, CA, February 17–20, 1991. p. 83–92, Paper SPE 21 209.
[51] Wang P, Balay S, Sepehrnoori K, Wheeler MF, Abate J, Smith B, Pope GA. A fully implicit parallel EOS compositional simulator for large scale reservoir simulation. In: Proceedings of Fifteenth SPE Symposium on Reservoir Simulation, Houston, TX, February 14–17, 1999. p. 63–71, Paper SPE 51885.
[52] Warren JE, Root PJ. The behavior of naturally fractured reservoirs. Soc Pet Eng J. p. 245–55; Trans SPE AIME 1963; 228.
[53] Wheeler MF, Arbogast T, Bryant S, Eaton J, Lu Q, Peszynska M, Yotov I. A parallel multiblock/multidomain approach for reservoir simulation. In: Proceedings of Fifteenth SPE Symposium on Reservoir Simulation, Houston, TX, February 14–17, 1999. p. 51–61, Paper SPE 51884.
[54] Wheeler JA, Smith RA. Reservoir simulation on a hypercube. SPE Reservoir Eng 1990;544.
[55] Wu YS. A virtual node method for handling wellbore boundary conditions in modeling multiphase flow in porous and fractured media. Water Resour Res 2000;36(3):807–14.
[56] Wu YS, Liu J, Xu T, Haukwa C, Zhang W, Liu HH, Ahlers CF. UZ flow models and submodels. Report MDL-NBS-HS-000006, Lawrence Berkeley National Laboratory Berkeley, CA; Las Vegas, Nevada, CRWMS M&O, 2000.
[57] Wu YS, Haukwa C, Bodvarsson GS. A site-scale model for fluid and heat flow in the unsaturated zone of Yucca mountain, Nevada. J Contam Hydrol 1999;38(1–3):185–217.
[58] Yevi G, Cinnella P, Zhuang X. On parallelizing a groundwater pollution simulators. Appl Math Comput 1998;89(1–3):313–25.

[59] Young LC, Hemanth-Kumar K. High-performance block oil computations. In: Proceedings of Eleventh SPE Symposium on Reservoir Simulation, Anaheim, CA, February 17–20, 1991. p. 135–44, Paper SPE 21215.

[60] Zhang K, Wu YS, Ding C, Pruess K. Application of parallel computing techniques to large-scale reservoir simulation. In: Proceedings of the 26th Annual Workshop, Geothermal Reservoir Engineering, Stanford, Stanford University, CA, January 29–31, 2001a.

[61] Zhang K, Wu YS, Ding C, Pruess K, Elmroth E. Parallel computing techniques for large-scale reservoir simulation of multi-component and multiphase fluid flow. In: Proceedings of the 2001 SPE Reservoir Simulation Symposium, Houston, TX, 2001b, Paper SPE 66343.

[62] Zhang H, Schartz FW, Sudicky EA. On the vectorization of finite element codes for high-performance computers. Water Resour Res 1994;30(12):3553–9.