SPE 106026

Efficient Parallel Simulation of CO_2 Geologic Sequestration in Saline Aquifers

Keni Zhang, Christine Doughty, Yu-Shu Wu, SPE, and Karsten Pruess, SPE, Lawrence Berkeley Natl. Laboratory

This paper was prepared for presentation at the 2007 SPE Reservoir Simulation Symposium held in Houston, Texas, U.S.A., 26–28 February 2007.

Abstract

An efficient parallel simulator for large-scale, long-term CO_2 geologic sequestration in saline aquifers has been developed. The parallel simulator is a three-dimensional, fully implicit model that solves large, sparse linear systems arising from discretization of the partial differential equations for mass and energy balance in porous and fractured media. The simulator is based on the ECO2N module of the TOUGH2 code and inherits all the process capabilities of the single-CPU TOUGH2 code, including a comprehensive description of the thermodynamics and thermophysical properties of H_2O-NaCl-CO_2 mixtures, modeling single and/or two-phase isothermal or non-isothermal flow processes, two-phase mixtures, fluid phases appearing or disappearing, as well as salt precipitation or dissolution. The new parallel simulator uses MPI for parallel implementation, the METIS software package for simulation domain partitioning, and the iterative parallel linear solver package Aztec for solving linear equations by multiple processors. In addition, the parallel simulator has been implemented with an efficient communication scheme. Test examples show that a linear or super-linear speedup can be obtained on Linux clusters as well as on supercomputers. Because of the significant improvement in both simulation time and memory requirement, the new simulator provides a powerful tool for tackling larger scale and more complex problems than can be solved by single-CPU codes. A high-resolution simulation example is presented that models buoyant convection, induced by a small increase in brine density caused by dissolution of CO_2.

Introduction

CO_2 geologic sequestration in saline aquifers and in oil and gas reservoirs involves complex multiphase flow processes as well as geomechanical, and geochemical processes, such as advection and diffusion, convective mixing, phase appearance/disappearance, dissolution and precipitation of minerals, and other chemical reactions. Mathematical models are effective tools for understanding the behavior of CO_2 in geological formations. Numerical modeling can in fact play an important role in evaluating the feasibility and reliability of CO_2 disposal. However, modeling of CO_2 geologic sequestration and migration processes in general requires fine spatial and temporal discretization, and represents a large computational challenge.

One of the popular numerical simulators for CO_2 geologic sequestration in saline aquifers is the ECO2N module of the general-purpose reservoir simulator TOUGH2[1-2]. The code can be used to model non-isothermal multiphase flows of water, salt, and CO_2 mixtures. TOUGH2/ECO2N represents fluids as consisting of two potentially mobile phases: a water-rich aqueous phase, a CO_2-rich gaseous phase, and an immobile solid halite phase. Because of the complexity of subsurface flow processes, most simulations for these types of problems are limited to systems of up to several ten thousand gridblocks. For reliable field scale applications, however, hundred thousands and millions of gridblocks may be needed to represent both geologic heterogeneities and multiphase, multicomponent flow structures on different scales.

In this study, a parallel simulator for large-scale, long-term CO_2 geologic sequestration in saline aquifers has been developed that is based on the ECO2N module of the TOUGH2 code. The TOUGH2 code itself was originally parallelized on CRAY T3E and IBM SP supercomputers[3-4], and later ported to Linux clusters and multi-core PCs[5]. The parallel version of TOUGH2, TOUGH_MP, has been successfully applied to solve multi-million gridblock multiphase fluid-flow problems and large-scale discrete fracture flow simulations[6-7]. The CO_2 parallel simulator retains all the process-modeling capabilities of the original TOUGH2/ECO2N and parallel computation features of TOUGH_MP. The parallel simulator is a three-dimensional, fully implicit model that solves large, sparse linear systems arising from discretization of the mass and energy balance equations in porous and fractured media. The simulator provides a comprehensive description of the thermodynamics and thermophysical properties of H_2O-NaCl-CO_2 mixtures, and models single and/or two-phase isothermal or non-isothermal flow processes, two-phase mixtures, appearance or disappearance of fluid phases, as well as salt precipitation or dissolution.

In this study, a domain decomposition approach is adopted for model parallelization and MPI (Message Passing Interface) for parallel implementation. The code partitions the simulation domain, defined by an unstructured grid, using a partitioning algorithm from the METIS software package[8]. In a parallel

simulation, each processor will treat one portion of the simulation domain for updating thermophysical properties, assembling mass and energy-balance equations, solving linear equation systems, and performing other local computations. Local linear-equation systems are solved in parallel by multiple processors with the Aztec linear solver package[9]. This parallel simulator has been built with an efficient communication scheme. Detailed discussion of the prototype of the data-exchange scheme can be found in Elmroth et al.[10], and improvements in the communication scheme are further discussed in Zhang and Wu[5].

The parallelization of TOUGH2 improves modeling capabilities significantly in terms of problem size and simulation time. The code demonstrates excellent scalability. Test examples show that a linear or super-linear speedup can be obtained on Linux clusters as well as on supercomputers. Because the parallel simulator was developed from an existing mature code, it inherits not only simulation functions from the original code, but also all other features, including input/output format, error handling, and improvements for code stability. These features provide robustness of the parallel code and ease of use for the user community of the original code, using identical input data, mesh and output files. Moreover, the domain decomposition approach and parallel computation enhance model simulation capabilities in terms of problem size and complexity to a level that cannot be reached by single-CPU codes. By using the parallel simulator, multi-million gridblock problems can be run on a typical Linux cluster with several tens to hundreds of processors to achieve ten to hundred times improvement in computational time or problem size.

Our tests indicate that the parallel simulator allows much larger problems to be solved by multiple-process simulation even with a single-processor CPU. This surprising result can be understood in terms of efficiency gains from decomposing one large linear algebra problem into a series of smaller ones, which produces super-linear speedup. The growing availability of multi-core CPUs will make parallel processing on PCs far more attractive, and parallel computing on multi-core PCs may soon become a standard for scientific computing as well as for engineering applications.

In this paper, we present a high-resolution simulation of buoyant convection, induced by a small increase in brine density caused by dissolution of CO_2. This example illustrates the multi-scale nature of convective mixing as an important process for the long-term fate of stored CO_2 in the subsurface.

Mathematical Model and Code Parallelization

Fluid Phases and Thermophysical Properties. The parallel simulator is designed to retain all the functionality of the original single-CPU TOUGH2/ECO2N[2]. ECO2N can represent a three-component system of water-CO_2-NaCl in two different fluid phases: an aqueous phase that is mostly water but may contain dissolved CO_2, and a CO_2-rich phase that may contain some water. All sub and super-critical CO_2 is considered as a single non-wetting phase. Salt may exist dissolved in brine and as solid precipitate.

In the numerical simulation of brine-CO_2 flows, the fundamental thermodynamic variables that characterize the brine-CO_2 system include system pressure, salt concentration, CO_2 concentration, and temperature. These four variables are considered as "primary variables" in defining the state of water-NaCl-CO_2 mixtures. The salt concentration variable can represent either mass fraction or solid saturation (fraction of pore space containing solid salt precipitate), depending on whether or not solid salt is present. Likewise, depending on phase composition, the CO_2 concentration variable can represent CO_2 mass fraction or saturation of the CO_2-rich phase. All other parameters, called secondary parameters, are computed from primary variables.

The partitioning of water and CO_2 among co-existing aqueous and gas phases is determined from a slightly modified version of the correlations developed by Spycher and Pruess[11]. These correlations were derived from the requirement that chemical potentials of all components must be equal in all phases. The equilibrium composition of liquid and gas phases is described as a function of temperature, pressure, and salinity, generally within experimental error, for the temperature range 12 ^{o}C to 110 ^{o}C, pressure up to 600 bar, and salinity up to full halite saturation. For the conditions of interest to CO_2 geological sequestration, equilibrium between aqueous and gas phases corresponds to a dissolved CO_2 mass fraction in the aqueous phase on the order of a few percent, and mass fraction of water in the gas phase of a fraction of 1%. Based on CO_2 mass fraction and CO_2 equilibrium phase composition, possible fluid phase conditions include single-phase liquid, single-phase gas, and two-phase fluid. ECO2N simulates phase transitions based on CO_2 mass fraction change. The equilibrium solubility of NaCl for typical sequestration conditions is around 25%. When salt mass fraction is larger than the equilibrium solubility of NaCl, solid salt will precipitate. Pruess[2] provides a more detailed discussion of the fluid phases and thermodynamic variables in the system of water-NaCl-CO_2 that ECO2N can simulate.

Mathematical Model. The TOUGH_MP code solves the same equation system as the original TOUGH2 code. The basic mass- and energy balance equations solved by TOUGH2 can be written in the general integral or finite volume form[1,12]:

$$\frac{d}{dt}\int_{V_n}\mathbf{M}^{\kappa}dV_n = \int_{\Gamma_n}\mathbf{F}^{\kappa}\bullet\mathbf{n}d\Gamma_n + \int_{V_n}q^{\kappa}dV_n \qquad (1)$$

The integration is over an arbitrary subdomain V_n of the flow system under study, which is bounded by the closed surface Γ_n. The quantity M appearing in the accumulation term (left hand side) represents mass or energy per volume, with κ= 1, ..., NK labeling the mass components (water, air, CO_2, solutes, ...; NK is the total number of mass components), and κ= NK + 1 the heat "component" **F** denotes mass or heat flux (see below), and q denotes sinks and sources. **n** is a normal vector on surface element $d\Gamma_n$, pointing inward into V_n.

Time and space discretization of Equation (1) using the integral finite difference (IDF) method results in a set of coupled non-linear equations, which can be written in residual form as follows[1]:

$$R_n^{\kappa}(x^{t+1}) = M_n^{\kappa}(x^{t+1}) - M_n^{\kappa}(x^t) - \frac{\Delta t}{V_n}\{\sum_m A_{nm} F_{nm}^{\kappa}(x^{t+1}) + V_n q_n^{\kappa,t+1}\} = 0 \qquad (2)$$

where the vector x^t consists of primary variables at time t, R_n^{κ} is the residual of component κ for grid block n, M denotes mass or thermal energy per unit volume for a component, V_n is the volume of the block n, q denotes sinks and sources of mass or energy, 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" term (including fluid flow, heat transfer, and advective and diffusive mass transport) between them. Equation (2) is solved by Newton/Raphson iteration, leading to

$$-\sum_i \left.\frac{\partial R_n^{\kappa,t+1}}{\partial x_i}\right|_p (x_{i,p+1} - x_{i,p}) = R_n^{\kappa,t+1}(x_{i,p}) \qquad (3)$$

where $x_{i,p}$ represents the value of i^{th} primary variable at the p^{th} iteration step.

Code Parallelization. The massively parallel simulation scheme implemented into the parallel CO_2 simulator is discussed in detail in Zhang et al.[3] and Wu et al.[4]. In the following sections, we will briefly summarize the most important steps and strategies in parallelizing the code. We also highlight recent improvements, including domain partitioning, parallel assembly of Jacobian matrix, parallel solution of linear equations, and message passing.

Domain Partitioning. An effective method for partitioning unstructured grids is a critical first step for a successful parallel scheme using the domain-decomposition approach. To obtain optimal parallel performance, the partitioning algorithm should take into account computational load and communication volume balance. In addition, the partition algorithm should also minimize communication volume in terms of message number and message size. It can be very difficult to achieve an ideal domain partitioning for an unstructured grid. To find an optimal selection and trade-off between these issues, we must take into account computer system characteristics, such as floating-point performance and bandwidth and latency of the communication subsystem. Because of the complexity of the problem, involving many variables, commonly used algorithms and software for partitioning large unstructured grids can not generally take all these issues into account. The current practice typically finds a trade-off between computation load balance and low total communication volume, even though this may not be theoretically optimal.

In a TOUGH2 simulation, a model domain is represented by a set of one-, two- or 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 gridblocks, an adjacency matrix can be constructed. The adjacency or connection structure of a model grid is stored in a compressed storage format (CSR).

We utilize the three partitioning algorithms in the METIS package (version 4.0)[8] for partitioning a grid domain. The three algorithms are here denoted as the *K-way*, the *VK-way*, and the *Recursive* partitioning algorithms. *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. The *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.

Only the nonzero entries of a submatrix for a partitioned mesh domain are stored in the processors. Each processor stores only the rows that are assigned to it. These rows form a submatrix whose entries correspond to variables of both the grid blocks assigned to the current processor, and to grid blocks assigned to neighboring processors that directly connect to the blocks defined in this processor. After partitioning, local numbering of gridblocks is carried out to facilitate the communication between processors. Through domain partitioning, the total memory requirement for a model simulation is distributed to processors involved in the computation. This guarantees effective use of computing and memory resources of all processors.

Parallel Assembly of Jacobian Matrix. In the TOUGH2/ECO2N formulation, the discretization of mass and energy conservation equations in space and time using the IFD method results in a set of strongly coupled nonlinear algebraic equations, which are solved by the Newton method. The Jacobian matrix needs to be calculated at each Newton iteration step, and this computational effort will be extensive for a large-scale simulation. In the parallel code, assembly of the Jacobian matrix is parallelized, and the work is shared by all the processors. Each processor is responsible for computing the rows of its own Jacobian matrix that correspond specifically to the processor's own gridblocks. Computation of the elements in the Jacobian matrix is performed in two parts. The first part consists of computations related to individual blocks (accumulation and source/sink terms). Such calculations are carried out using the information stored in the current processor; no communications with other processors are needed. The second part includes all computations relating to the connections or "flow" terms. Calculation of flow terms for gridblocks at the sub-domain border requires information from the neighboring sub-domain, which in turn requires communication with neighboring processors. Before performing these computations, there must be an exchange of relevant primary variables across different model sub-domains.

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[13]. All matrix blocks are stored row-wise, with the diagonal blocks stored first in each block row. Scalar elements of gridblocks are stored in a column major order. The data structure consists of a real type vector and five integer type vectors, forming the Jacobian

matrix. Detailed explanation of the DVBR data format can be found in Tuminaro et al.[9].

Parallel Solution of Linear Equations. The linear equation system arising at each Newton step is solved using an iterative parallel linear solver from the Aztec package[9]. Different solver options and preconditioners from the package may be selected. The available solvers include conjugate gradient, restarted generalized minimal residual, conjugate gradient squared, transposed-free quasi-minimal residual, and bi-conjugate gradient with stabilization methods. Again, the work of 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, time steps are automatically adjusted (increased or reduced) depending on the convergence rate of Newtonian iterations. In the parallel code, the time-step size is calculated at the master processor 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 Among Processors. Communication of data between processors working on neighboring/connected gridblocks, partitioned into different domains, is an essential component of the parallel algorithm. Global communication is required for checking convergence, collecting extreme values, conditioning on whole simulation domain, and input/output contributed by all processors. In addition to the communication taking place inside the Aztec routine to solve the linear equation system, communication between neighboring processors is necessary to update primary variables.

It is possible to reduce the communication volume for boundary message exchange at the end of each Newton iteration by introducing some computation overlap. Note that TOUGH2 uses two types of variables: the primary thermodynamic variables for all grid blocks, and all other (secondary) thermophysical parameters needed to assemble the governing flow and transport equations. At each iteration step, primary variables are solved directly from the equation system, and secondary variables are then updated based on the new primary variables. To reduce communication volume, boundary message exchange is limited to the primary variables only. Entries of the Jacobian matrix related to the gridblocks with at least one connection to a gridblock assigned to another processor require values from other processors to be updated. The set of gridblocks not in the current processor, but needed to update components in the border gridblocks of the current processors, is referred to as an *external* set. If only the primary variables are communicated, secondary variables for the *external* set must be updated at the current processor, using the most updated primary variables of the *external* set. This approach will introduce an additional computational burden for updating secondary variables for the *external* set. However, such trade-off between computation and communication has shown better efficiency for the code on most Linux clusters.

Small size messages are needed to coordinate the processors involved in parallel simulations. Reducing the total number of small messages is an important factor for improving code performance, especially on a high-latency computer system. The most frequent small messages may include messages for finding and collecting information. The code finds global extreme values, performs global conditioning, and calculates global norms through reducing and gathering operations. The number of small messages may be reduced through combining of several messages into one message. For example, at the end of each Newton iteration step, the code needs to check convergence of the iteration. This check requires collecting the maximum residual from all processors. If the maximum residual is less than the convergence criterion, the program will progress to the next time step. The code also needs to gather information about the origin of the maximum residual (from which gridblock and which component). Different types of messages (real, character, and integer) may be combined to one small array for communication. Another frequent message involves collecting values of a variable from all processors. Then, based on the variable values, the code will perform different actions. If the action is to stop program execution, communication may not be necessary. The program can directly examine conditions for the variable values at local processors. If the conditions are satisfied, the processor will inform all other processors to stop execution.

Investigation of Parallel Performance

The efficiency of the parallel simulator depends on two factors: computation and communication. Because we use computation schemes in the parallel code that are similar to the original code for assembling the Jacobian matrix, updating thermophysical parameters, and Newtonian iteration, efficiency of the code is mainly determined by the efficiency of communication and the parallel linear solver. Through domain decomposition, Jacobian matrix assembly, thermophysical property update, and Newton iterations are fully parallelized with a small amount of overlap. However, domain partitioning schemes used in the parallel code may introduce additional non-linearity to the global discretized equations for describing a fully coupled physical system, due to handling flux terms between adjacent partitioned grid domains. This additional perturbation to the equation system from parallelization may partially cancel out the numerical performance benefit from parallel computing itself when an extremely large number of processors is used.

We use a field-scale problem to verify the parallel simulator and to test its performance. In particular, simulation results from the original TOUGH2/ECO2N and the parallel simulator are compared to demonstrate that the new simulator gives correct solutions. The performance of the new simulator is then examined by running the model using different numbers of processors on different platforms.

The problem involves injection of supercritical carbon dioxide (CO_2) into a highly permeable, steeply dipping saline aquifer. It is based on the Frio brine pilot, a research program conducted at the South Liberty field, Dayton, Texas, where

CO_2 was injected at a variable rate, averaging about 1.85 kg/sec (160 metric tons per day) for ten days[14]. The saline aquifer in the upper Frio Formation is 23 m thick, and is laterally compartmentalized by faults. The fault block modeled is about 1 km wide, and over 2 km long (see Figure 1). It dips at an angle of 16° in the long direction. The model has 13 layers, each containing 3,596 gridblocks. Layers are tilted at an angle of 16° from the horizontal to represent the average formation dip. Porosity and permeability vary by layer, and are taken from wireline logs correlated to core-sample analysis. Porosity averages 0.34 and permeability averages about 2,500 md. At ambient conditions (pressure of 150 bars, temperature of 56 °C, and salinity of 75,000 ppm TDS), the injected CO_2 is supercritical. The injection well is perforated in the upper 6 m of the formation, as is the one observation well located about 30 m in the updip direction from the injection well.

Simulations shown here maintain isothermal conditions (the energy equation is not solved) and cover the first four days of injection. Because the simulations consider only the injection of CO_2, drainage is the dominant flow process and non-hysteretic characteristic curves are used. Relative permeability is modeled using Corey curves[15] with an irreducible liquid saturation of 0.2 and a near-zero residual gas saturation. Capillary pressure is modeled using the van Genuchten formulation[16] with parameters chosen to match laboratory mercury-intrusion experiments on core samples from the injection well.

The model was run with the same input files, using both the parallel and the original sequential codes. Figure 2 shows the comparison of simulated pressures and gas saturations in the injection and observation wells. The comparison indicates that both codes produce nearly identical results for this complex three-dimensional (3D) field scale problem. Startup of CO_2 injection causes pressures and gas saturation to rise initially at these two locations. Later changes in pressure and gas saturation occur due to injection rate changes. Figure 3 shows the CO_2 mass fraction distribution after four days of injection. The figure indicates that the CO_2 has spread over about a 40 m range in 4 days.

Code Performance on Linux Clusters. Simulations were conducted on a 10-node Linux cluster, with each node equipped with two INTEL Xeon 3.60 GHz CPUs. The computation times for a four-day simulation run using different numbers of processors are shown in Table 1. The simulation was also run using the original TOUGH2/ECO2N code using one processor, its computation time is included in Table 1 for comparison.

The computation time is reduced significantly as the number of CPUs increased. In general, performance of the code for the two parts, (1) updating thermophysical parameters and assembling Jacobian matrix, and (2) solving linear equations, shows a linear or super linear speedup, as does total execution time. The parallel code can gain about 20% time saving using two processors compared to using one processor with the original sequential code for this problem. Total execution time is reduced to less than half when doubling the number of processors, except from 2 processors to 4 processors. The exception for 2 processors to 4 processors is because the 2 processors reside inside one node with shared memory and 4 processors are from two nodes which require additional time for cross-node communication.

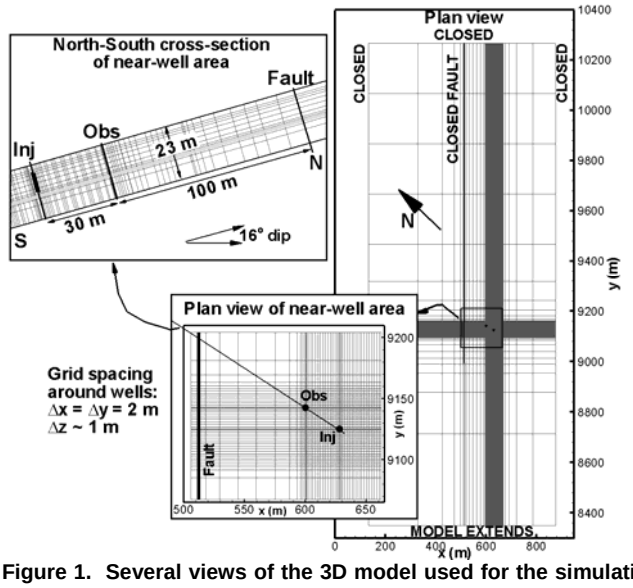

Figure 1. Several views of the 3D model used for the simulations of CO_2 injection at the Frio brine pilot.

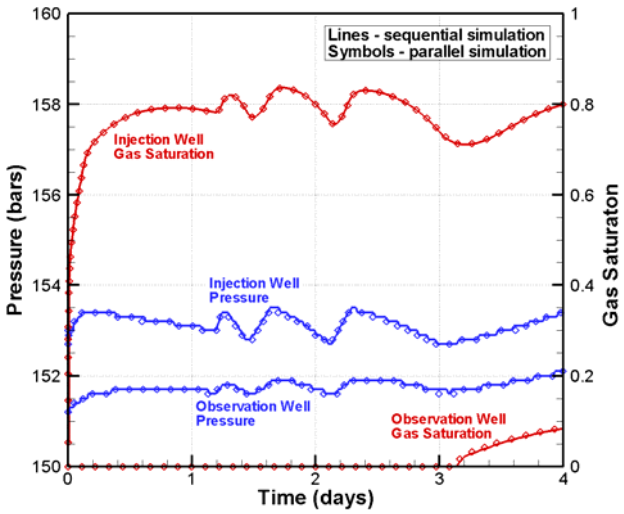

Figure 2. Comparison of simulated CO_2 gas saturation and pressure at the injection and observation wells of the Frio brine pilot by the two codes.

Table 1. Comparison of execution times for the simulation using different numbers of processors

Number of CPUs	1*	2	4	8	16
Time for updating thermophysical parameters and assembling Jacobian matrix (s)		1794	933	504	262
Time for solving linear equations (s)		11132	5764	2798	1220
Total execution time (s)	15480	12929	6700	3307	1485
Relative Speedup**	1.00	1.20	1.93	2.03	2.23

** running the original (non-parallel) TOUGH2/ECO2N code.*

*** ratio of total execution times between current case and previous case with half as many CPUs*

Note that the total execution time also includes time for input/output, initialization and others. Table 1 shows that parallelization of the linear-equation solution achieves much better than linear speedup. Elmroth et al.[10] provided a theoretical analysis for this very favorable phenomenon and indicated that the super linear behavior results mainly from by the performance of the preconditioner.

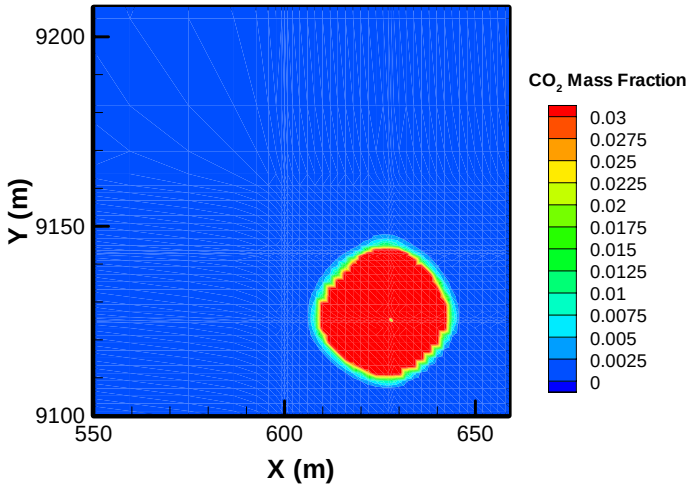

Figure 3. Plan view of the CO_2 mass fraction distribution at the injection level (-6.55 m) after four days of injection for the Frio brine pilot.

Speedup on Multi-core PCs. The super linear speedup phenomenon was further investigated on a dual-core PC with a Linux operating system. The parallel simulator was run on the PC with a number of processes specified that was larger than the actual CPU number of the PC. With an increase in the number of processes running on the two CPUs, computation time was found to reduce slightly. For most problems, a saturation point will be reached at 10-12 processes. Beyond the saturation point, increasing the number of processes will lead to longer computation times. This is because the coordination and communication requirements for multiple processes will override the benefit of super liner speedup obtained by domain decomposition. It is important to note that multi-process speedup is obtained only when the problem size is large. In general, a simulation problem should have at least several ten-thousand gridblocks for achieving benefits from this approach. We have obtained an additional 20-30% speedup when 8 processes run on two CPUs, as compared to running 2 processes on the two CPUs. For larger problems (more gridblocks) better speedup can be obtained.

Parallel computing on multi-core PCs becomes very promising with the development of multi-core processors. Our tests indicate that much larger problems can be solved by multiple process simulation with the parallel simulator than is possible with single process simulation using the original ECO2N/TOUGH2 code. This may be because solving a large problem involves more overhead with the traditional reservoir simulators than by parallel solving several smaller problems for the same problem. In addition, one of the possible limitations for doing large-scale simulations on PCs is computer memory size. We have successfully run a million-gridblock model on a dual-core PC with a memory of 2 GB. Abundant PC computing resources and a robust parallel code make million gridblock simulations possible to be performed on PCs.

The benefits from using a domain decomposition approach that runs more processes than available number of CPUs may depend on the computer system, such as operating system, MPI installation, and hardware. Further studies are needed.

Large-scale simulation: Investigation of CO_2 Convection Mixing

When CO_2 is injected into a saline formation, it partially displaces the resident brine, and partially dissolves in it, while some water also dissolves (evaporates) into the flowing CO_2 stream. Under most subsurface temperature and pressure conditions, CO_2 is buoyant (less dense), compared to water (or brine), and the injected CO_2 will move upward towards the top of the permeable interval. Eventually, the carbon dioxide is distributed among mobile layers beneath the caprock. One of the interesting phenomena affecting CO_2 migration processes is convective mixing. When CO_2 dissolves in brine, the density of the aqueous phase will increase by a small amount of approximately 1 %. Although small, the density increase is sufficient to generate convection cells in the formation, provided there is "sufficient" vertical permeability. Some initial work has been done to demonstrate such convection in two-diemnsional space[17-18], and to show that the time scales for significant effects may be long (hundreds to thousands of years). While free CO_2 is buoyant and always poses a threat of upward escape from storage, its dissolution in the aqueous phase and subsequent buoyant convection has the potential of placing CO_2 under conditions that restrict its tendency for upward migration. In addition, and most importantly, the convection process may distribute and mix dissolved CO_2 throughout the permeable thickness of the storage formation, potentially providing a far greater storage volume than could be accessed by the free-phase CO_2.

CO_2 convection starts slowly and within a small space scale, which over time grows to larger-scale flows. We are using a three-dimensional high-resolution model to begin a systematic evaluation of the role of brine convection in enhancing CO_2 dissolution. Figure 4 shows a schematic of phase distributions around a typical CO_2 injection well and the 3D model domain. Our initial studies used a cube of 1m×1m×1m size to investigate the onset and early stage of CO_2 convection. Fine gridding (Δx = 1 cm) is used for representing the interplay between molecular diffusion and aqueous phase convection induced by the small density change due to CO_2 dissolution, resulting in 100x100x100 = 1,000,000 gridblocks. With an additional 10,000 top boundary gridblocks, the model includes 1,010,000 gridblocks and 2,999,800 connections between them.

Model initial and boundary conditions are shown in Figure 5. Two-phase conditions with a free CO_2-rich phase (gas saturation S_g = 0.1 %) are maintained at the upper boundary. Conditions of no fluid flow at the upper boundary are enforced by specifying a very small permeability in the boundary domain (10^{-50} m^2). This results in mass transport across the top boundary occurring by molecular diffusion only. The model domain is assigned an isotropic permeability of 10^{-11} m^2.

Permeabilities for all gridblocks are modified by multiplying the assigned permeability with a random number in the range 0.99-1.01. This small modification is applied to trigger the onset of CO_2 convection.

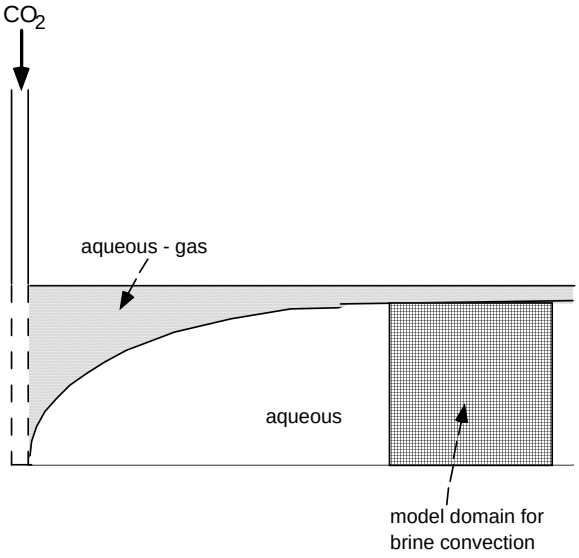

Figure 4. Schematic of phase distributions around a CO_2 injection well. A subdomain used for brine convection modeling is also shown.

The flow domain is initialized with a constant pressure, but because of the small compressibility of the aqueous phase, hydrostatic pressure equilibrium is established virtually instantaneously. CO_2 then diffuses into the initially CO_2-free aqueous phase below, causing brine density to increase and eventually triggering downward advection. The unstable nature of the advection (denser fluid above less-dense fluid) gives rise to fingering, see Figures 6-9.

Figure 6 shows CO_2 mass fractions at a time of 8.53 days at x=0.505 m, which represents a vertical cross-section located near the center of the cube. In this case, pure water is used (no salinity). Figure 7 demonstrates convective fingering in water with 12.5% salinity. The figure shows the CO_2 mass distribution at time 20.24 days. Even with a longer dissolution period, the fingering is much weaker than in the pure water case. This is because the higher salinity reduces CO_2 solubility, weakens the driving force for convection, and delays finger development. Figure 8 further illustrates the salinity influence on the development of fingers. In the brine with 25% salt, no convection has developed after 115.7 days. However, with higher permeability, fingers will be formed more easily. Figure 9 shows the CO_2 mass distribution for the 25 % salinity case with permeability increased by a factor of 18.7. In this case, fingers develop, but with lower mass fraction.

Additional simulation results and further discussions will be published elsewhere.

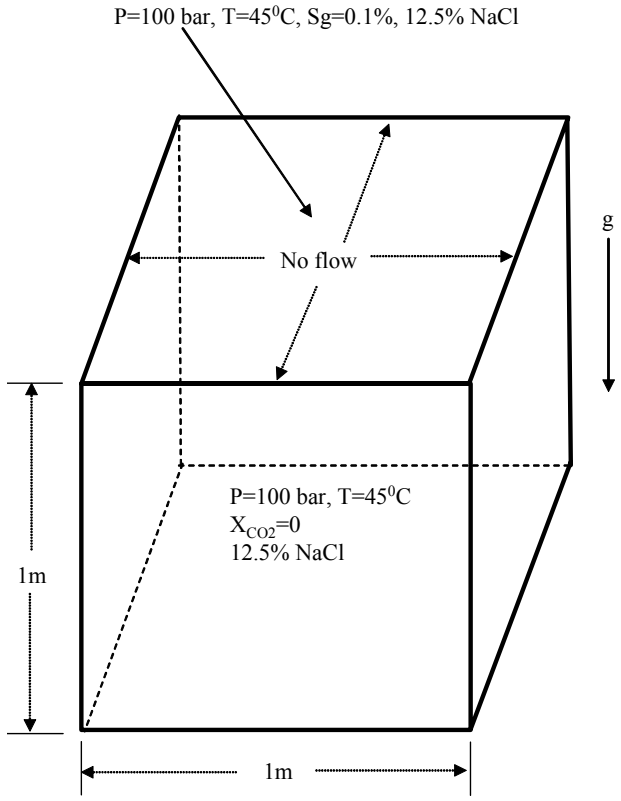

Figure 5. Three-dimensional domain for simulating brine convection induced by CO_2 dissolution and associated increase in aqueous phase density. Initial and boundary conditions are also shown.

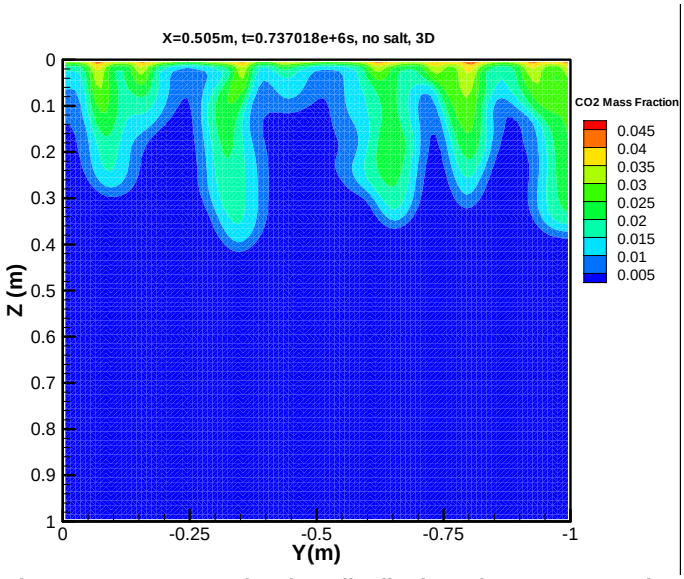

Figure 6. CO_2 mass fraction distribution along cross-section x=0.505 m at time 8.53 days, NaCl=0.0 in the water.

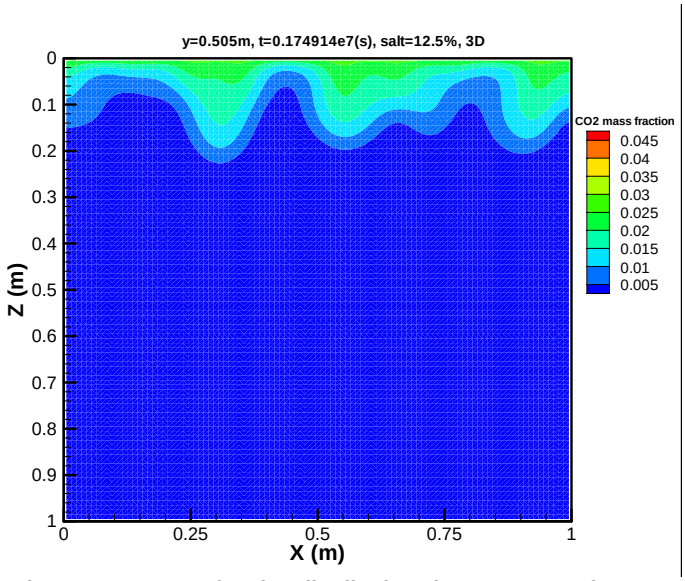

Figure 7. CO_2 mass fraction distribution along cross-section y=0.505 m at time 20.2 days, NaCl= 12.5 % in the water.

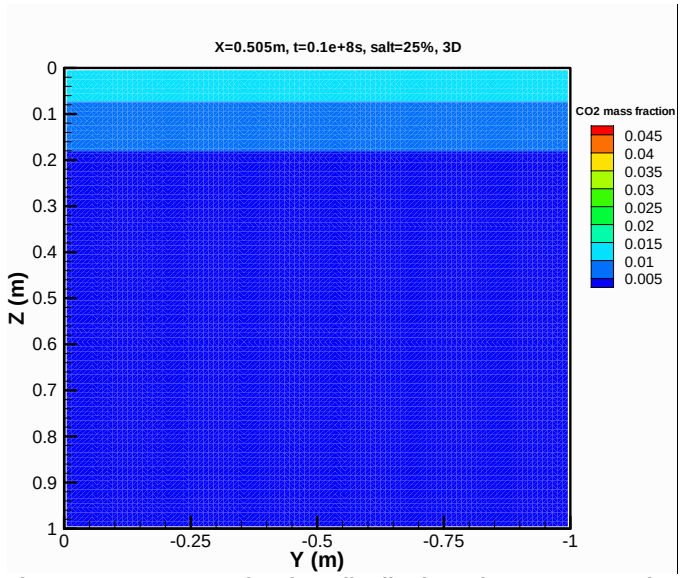

Figure 8. CO_2 mass fraction distribution along cross-section x=0.505m at time 115.7 days, NaCl= 25.0 % in the water.

Code performance for the 3-D Problem. The above simulations were run on a Linux cluster at NERSC (National Energy Research Scientific Computing Center). The cluster is equipped with 356 nodes using infiniBand switch connection, and each node consists of 2 Opteron 2.2 GHz CPUs. For testing the parallel code performance, the case of 12.5 % salinity was run using either 2, 4, 8, 16, 32, 64, 128, or 256 processors. Figure 10 shows the speedups obtained for different numbers of processors and for different parts of the simulation. By increasing the number of processors, the total execution time was reduced from 79,160 seconds using two processors to 514 seconds using 256 processors. The parallel code demonstrates much better performance than ideal linear speedup. The total execution time is reduced to less than half when doubling the processor numbers when using 64 or less processors for this problem. Figure 10 indicates that the super-linear effect is introduced by the linear equation solution. It is interesting to note that this large-scale problem can be run on two processors, because of the huge memory available for each node (2 processors share 6 GB memory). This may further confirm the possibility of performing large-scale simulations on multi-core PCs, as discussed in Section 3.2.

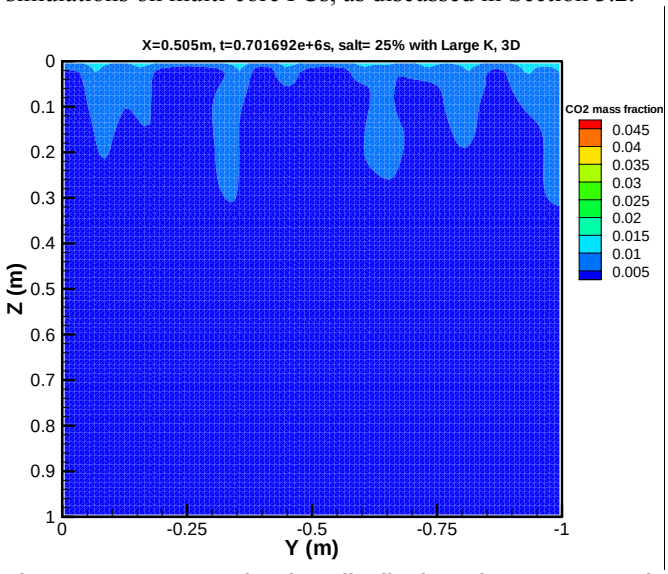

Figure 9. CO_2 mass fraction distribution along cross-section x=0.505 m, NaCl= 25.0 % in the water, with an increased permeability (factor 18.7).

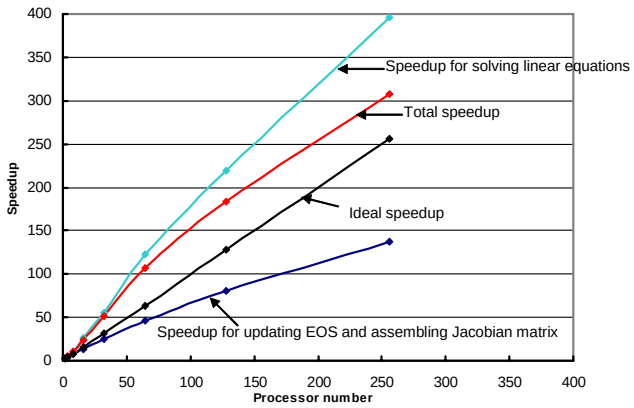

Figure 10. Speedups for the different parts of the parallel simulations (CO_2 Convection, Salinity=12.5 %).

Conclusions

An efficient parallel simulator for large-scale, long-term CO_2 geologic sequestration in saline aquifers has been developed based on the ECO2N module of the general-purpose numerical simulation program TOUGH2. This parallel simulator is a three-dimensional, fully implicit model that solves large, sparse linear systems arising from discretization of the partial differential equations for mass and energy balance in porous and fractured media. The simulator retains all the process-modeling capabilities of the single CPU code TOUGH2/ECO2N. The parallel code is shown to be computationally efficient.

The parallel simulator provides a powerful tool to tackle larger scale and more complex problems than can be solved currently by sequential codes. It is expected that the new simulator will enhance modeling capacity in terms of model

size and simulation time by 1-3 orders of the magnitude. The code is designed to be easy to use and little learning is necessary for experienced TOUGH2/ECO2N users.

The parallel simulator is used to investigate convective mixing induced by a small increase in brine density due to dissolution of CO_2. The 3D high-resolution model involves more than 1 million gridblocks, which is currently impossible to run without using a parallel simulator. The large-scale simulations provide insight into CO_2 convective mixing. The model results illustrate the multi-scale nature of convective mixing as an important process for the long-term fate of stored CO_2.

Acknowledgments

The authors would like to thank Lehua Pan and Dan Hawkes for their review of this paper. This work was supported by the Zero Emission Research and Technology project (ZERT) under Contract No. DE-AC02-05CH11231 with the U.S. Department of Energy. Computations were carried out partially on computer facilities provided by the NERSC.

References

1. Pruess, K, Oldenburg, C. and Moridis, G.: "TOUGH2 User's Guide, V2.," Lawrence Berkeley National Laboratory Report LBNL-43134, Berkeley, CA, 1999.
2. Pruess, K.: "ECO2N: A TOUGH2 Fluid property module for mixtures of water, NaCl, and CO_2," Lawrence Berkeley National Laboratory Report LBNL-57952, Berkeley, CA, 2005.
3. Zhang, K., Wu, Y.S., Ding, C., Pruess, K., and Elmroth, E.: "Parallel computing techniques for large-scale reservoir simulation of multi-component and multiphase fluid flow", Paper SPE 66343, Proceedings of the 2001 SPE Reservoir Simulation Symposium, Houston, Texas, 2001.
4. Wu, Y.S., Zhang, K., Ding, C., Pruess, K., Elmroth, E., and Bodvarsson, G.S.: "An efficient parallel-computing scheme for modeling nonisothermal multiphase flow and multicomponent transport in porous and fractured media," *Advances In Water Resources*, (2002)25, 243-261.
5. Zhang, K. and Wu, Y.S.: "Enhancing Scalability and Efficiency of the TOUGH_MP for Linux Clusters," proceedings of TOUGH Symposium 2006, Berkeley, CA 2006.
6. Zhang, K., Wu, Y.S. and Bodvarsson, G.S.: "Parallel computing simulation of fluid flow in the unsaturated zone of Yucca Mountain, Nevada," *Journal of Contaminant Hydrology*, (2003) 62-63, 381-399.
7. Zhang, K., Wu, Y.S., Bodvarsson, G.S., and Liu, H.H.: "Flow Focusing in Unsaturated Fracture Networks: A Numerical Investigation," Vadose Zone Journal, (2004) 3:624-633
8. Karypsis, G. And 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.
9. Tuminaro, R.S., Heroux, M., Hutchinson, S.A. and Shadid, J.N.: "Official Aztec user's guide, Ver 2.1," Massively Parallel Computing Research Laboratory, Sandia National Laboratories, Albuquerque, NM, 1999.
10. Elmroth, E., Ding, C., and Wu, Y.S.: "High performance computations for large-scale simulations of subsurface multiphase fluid and heat flow," *The Journal of Supercomputing*, (2001)18(3), pp. 233-256.
11. Spycher, N. and Pruess K.: "CO2-H2O Mixtures in the Geological Sequestration of CO2. II. Partitioning in Chloride Brines at 12–100 °C and up to 600 bar," *Geochim. Cosmochim. Acta,* (2005)69, No. 13, pp. 3309–3320, doi:10.1016/j.gca.2005.01.015.
12. Pruess, K.: "The TOUGH Codes—A Family of Simulation Tools for Multiphase Flow and Transport Processes in Permeable Media," *Vadose Zone J.*, (2004)3, pp. 738 - 746.
13. Carney, S., Heroux, M. and Li, G.: "A proposal for a sparse BLAS toolkit." Technical report, Cray Research Inc., Eagen, MN, 1993.
14. Hovorka, S.D., Benson, S.M., Doughty, C., Freifeld, B.M., Sakurai, M., Daley, T.M., Kharaka, Y.K., Holtz, M.H., Trautz, R.C., Nance, H.S., Myer, L.R. and Knauss, K.G.: "Measuring permanence of CO_2 storage in saline formations: the Frio experiment," *Environmental Geosciences*, (2006) 13(2), 1-17.
15. Corey, A.T.: "The Interrelation Between Gas and Oil Relative Permeabilities," *Producers Monthly*, (1954)11, pp. 38 - 41.
16. van Genuchten, M.Th.: "A Closed-Form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils," *Soil Sci. Soc. Am. J.*, (1980) 44, pp. 892 - 898.
17. Ennis-King, J. and Paterson, L.: "Role of Convective Mixing in the Long-Term Storage Dioxide in Deep Saline Formations", In: SPE annual technical conference and exhibition, SPE 84344. Denver, Colorado, 2003.
18. Xu, X., Chen, S., and Zhang, D.: "Convective Stability of the Long-term Storage of Carbon Dioxide in Deep Saline Aquifers", *Advances In Water Resources*, (2006)29,397-407.