SPE-173208-MS

Efficient Multi-Stage Preconditioners for Highly Heterogeneous Reservoir Simulations on Parallel Distributed Systems

Hui Liu, Kun Wang, and Zhangxin Chen, University of Calgary; Kirk E. Jordan, IBM

This paper was prepared for presentation at the SPE Reservoir Simulation Symposium held in Houston, Texas, USA, 23–25 February 2015.

Abstract

Large-scale reservoir simulation is still a big challenge due to the difficulty of solving linear systems resulted from the Newton methods. For black oil simulation, more than 90% of running time is spent on the solution of linear systems. The problem is getting worse when developing parallel reservoir simulators using parallel distributed systems with tens of thousands of CPUs. Efficient linear solvers and preconditioners are critical to the development of parallel reservoir simulators.

Here we introduce our work on developing parallel preconditioners for highly heterogeneous reservoir simulations. A family of new Constrained Pressure Residual (CPR)-like preconditioners and advanced matrix pre-processing techniques are developed, including two new three-stage preconditioners and one four-stage preconditioners. A pressure system is solved by an algebraic multi-grid method, and the entire linear system is solved by the restricted additive Schwarz (RAS) method (one of the domain decomposition methods). To overcome a convective issue in reservoir simulation, a parallel potential-based matrix reordering method is employed to stabilize our preconditioners. Matrix decoupling methods, such as an alternative block factorization (ABF) strategy and a Quasi-IMPES (implicit pressure explicit saturation) strategy, are also applied. With the restricted additive Schwarz and algebraic multi-grid methods, our preconditioners have good scalability for parallel computers.

Our preconditioners have been applied to oil-water and black oil benchmark simulations. For the SPE 10 project, which is a big challenge for a linear solver because of highly heterogeneous permeability and porosity, our preconditioners with the GMRES linear solver are stable and efficient. When using 64 CPUs, the number of iterations of our linear solvers is less than 40. When applying our method to a standard black oil simulation with 100 millions of grid blocks, the number of iterations of our linear solvers is only two using 3,072 CPU cores. Our numerical experiments show that our preconditioners and linear solvers are stable with a large number of CPUs and are efficient for highly heterogeneous simulations.

Introduction

The black oil model is a classical multi-phase flow model in reservoir simulations, which assumes that three components exist, including the water, oil and gas components, and three phases coexist in a reservoir. This model has been widely used in the petroleum industry. Now, many commercial simulators have been developed to simulate the black oil model, most of which are designed for workstations and

personal computers. These simulators are not efficient when simulating cases with millions of grid blocks and highly heterogeneous geological conditions. One reason is that the linear systems arising from the Newton or Newton-Raphson methods are extremely difficult to solve. The situation is becoming even worse when we run cases with tens of millions of grid blocks. Efficient linear solvers and preconditioners are fundamental to the development of parallel reservoir simulators.

Linear solvers and preconditioners are essential to most scientific computing applications and have been studied for decades. Various linear solvers and preconditioners have been proposed. Saad and Schultz developed the GMRES (generalized minimal residual method) solver [13, 1, 46, 66], which is a general purpose linear solver and works for all non-singular linear systems. This solver is the most widely used linear solver for non-symmetric linear systems, and it is the default solver used in many reservoir simulators, such as CMG and Eclipse. The ORTHOMIN (orthogonal minimum residual) and BICGSTAB (biconjugate gradient stabilized method) solvers are also efficient Krylov subspace solvers, work for general matrices [13, 1] and are applied in reservoir simulators. The conjugate gradient solver works for symmetric positive-definite matrices [13]. However, for linear systems from elliptic and parabolic problems, multigrid solvers are the most efficient solvers [44, 48, 49], including geometric multigrid (GMG) solvers and algebraic multigrid (AMG) solvers. The GMG solvers are suitable for applications with structured grids while the AMG solvers are more flexible, which work for any types of grids. For the black oil model, its pressure equation is parabolic or elliptic, and the AMG solvers have been applied to solve the linear systems arising from the discretization of this equation [16, 17, 23, 38, 45].

Preconditioners are essential to the success of linear solvers. The incomplete LU factorization (ILU) preconditioner is a common one for general linear systems [1, 14, 15, 46], which is also the basis of other preconditioners, such as domain decomposition methods. The ILU methods include ILU(k), ILUT(p, tol) and other forms [46, 66]. The domain decomposition (DDM) methods [26] are broadly used in parallel computing, one of which is the famous Restricted Additive Schwarz (RAS) method developed by Cai et al. [28, 30]. The RAS method uses less communication between subdomains than other domain decomposition methods and has better convergence [28]. The ILU and DDM preconditioners are general preconditioners, which work for any kinds of scientific applications. However, when simulating a specific problem, problem-related (i.e., physics-based) preconditioners usually have better performance than these general ones. Wallis et al. developed the constrained pressure residual (CPR) preconditioner for the black oil model [16], which is much more efficient than the ILU preconditioners and has been applied by many researchers to different reservoir simulators [17, 23, 38, 45]. New preconditioners based on the CPR preconditioner were also developed [17, 23, 38]. Cao et al. implemented a general multi-stage CPR-like preconditioner for the black oil model. Chen et al. also implemented a CPR preconditioner on the NVIDIA Tesla GPU [45, 35, 36], where point-wise ILU and block-wise ILU preconditioners were supported. To speed the solution of linear systems, matrix preprocessing techniques were studied, including matrix decoupling and matrix reordering. Matrix decoupling techniques weaken the coupling between unknowns. The commonly used decoupling techniques include the alternate-block-factorization strategy [18], Quasi-IMPES [20] and other strategies [22, 17, 38]. Matrix reordering techniques change the order of unknowns. The commonly used techniques include MD, AMD [13], RCM and potential-based reordering strategies [47]. The potential-based reordering method was developed for reservoir simulations, which works for the black oil, compositional and thermal models. More materials for preconditioners and matrix decoupling and reordering methods can be read in references [13, 1, 47].

Parallel computing techniques have been also adopted by the petroleum industry [2, 3, 4, 8]. Rutledge et al. [6] implemented a parallel compositional simulator on massive SIMD computers using the IMPES method. Killough et al. [5] developed a parallel compositional simulator, which demonstrated that a highly efficient parallel model can be generated for an n-component EOS reservoir simulator on a distributed-memory parallel computer. Shiralkar et al. [7] developed a parallel production quality simulator, which ran effectively on a variety of computing platforms. Parashar et al. implemented a parallel simulator,

which could handle multiple fault blocks with multiple physics [12]. Dogru et al. [11] developed a parallel simulator, which was capable of simulating one billion cells. Zhang et al. developed a parallel platform, PHG (Parallel Hierarchical Grid), for adaptive finite element methods and finite volume methods [58, 60, 62, 51], which was also used to develop parallel simulators. Recently, high performance GPUs, such as NVIDIA Tesla K10 and K20x, were also applied to accelerate reservoir simulations. Chen et al. developed GPU-based linear solvers and preconditioners to speed the Krylov subspace linear solvers, algebraic multigrid solvers and preconditioners [34, 35, 36, 37, 45, 65, 31, 39, 40, 41]. These techniques have been applied to the black oil model [35, 36, 65, 42, 43]. Klie et al. implemented their GPU-based linear solver package and applied it to reservoir simulations [32, 33]. OpenMP and MPI based parallel reservoir simulators have been commercialized. Meanwhile, more models have been considered, such as thermal, compositional, polymer flooding and chemical reaction models [46, 66, 19, 21, 24, 27, 29, 55, 56, 59, 61, 63, 42, 43].

This paper presents a family of CPR-like preconditioners, including CPR-FP, CPR-FPF, CPR-PFP, and CPR-FFPF. The F means solving the whole system using the Restricted Additive Schwarz (RAS) method. The P means solving a pressure submatrix using an algebraic multigrid (AMG) method. These preconditioners are developed for oil-water, water-gas, black oil, polymer flooding and extended black oil models. They are suitable for both sequential and parallel reservoir simulators. Here the RAS method is used to replace the ILU method in the classical CPR preconditioner. The pressure submatrix is solved by AMG from the HYPRE [48, 49], a world famous parallel AMG solver package. Numerical experiments are carried to test the efficiency of these preconditioners. The benchmarks show that our preconditioners are stable, efficient and scalable, and linear solvers using these preconditioners can converge in a few iterations.

Preconditioners for the Black Oil Model

The black oil model is highly nonlinear. Properties related to saturations are strongly nonlinear while properties related to pressures are weakly nonlinear. After discretization, the black oil model can be solved by the standard Newton, Newton-Raphson or inexact Newton methods [25]. Then a linear system

$$Ax = b \tag{1}$$

is derived from each nonlinear iteration, where A is an $N \times N$ non-singular matrix, b is a residual and x is the solution to be calculated. For the linear systems from reservoir simulations, regular Krylov subspace solvers are employed, including GMRES(m), BICGSTAB and ORTHOMIN. To speed these solvers, in practice, the following equivalent linear system is solved:

$$M^{-1}Ax = M^{-1}b, \tag{2}$$

where M is a left-preconditioner or preconditioner.

The equations corresponding to the pressure unknowns from the black oil model are elliptic or parabolic. In addition, the solution of the pressure equations dominates the overall error when solving the entire linear system $Ax = b$. For linear systems from elliptic or parabolic problems, the algebraic multigrid (AMG) methods are much more efficient than the ILU methods. Wallis et al. used the AMG methods to solve the pressure matrix and the ILU methods to solve the whole linear system. Based on the CPR preconditioner, many similar preconditioners have been developed. In this section, a family of CPR based preconditioners will be proposed. Before we go into details, some notation is introduced.

Notation. Depending on the operation conditions, the black oil model has different states. If the oil phase pressure is greater than the bubble point pressure, then there is no free gas, which is noted as the undersaturated state, and if the oil phase pressure is less than the bubble point pressure, gas, oil and water coexist in the system, and the state is called the saturated state. When discretizing the black oil model,

different primary variables (unknowns) may be chosen [46, 66]. The oil phase pressure p_o, the water saturation S_ω and the gas saturation S_g may be chosen for the saturated state. For the undersaturated state, since S_g is zero, the oil phase pressure p_o, the water saturation S_ω and the bubble point pressure p_b may be chosen as the primary variables. For the simplified two-phase oil-water model, the oil is dead oil, which means that there is no gas and solution gas in the system. The primary variables can be chosen to the oil phase pressure *po* and water saturation s_ω. Usually the oil phase pressure p_o is always one of the primary variables.

Assuming that a grid is chosen, which has n grid blocks, and the grid blocks are numbered from 1 to n. For each grid block i ($1 \leq i \leq n$), there are three primary variables, one of which is the oil phase pressure $p_{o,i}$. The other variables are denoted by $\vec{x_i}$. Depending on the states, $\vec{x_i}$ may contain one or two variables. For example, when working on the two-phase oil-water model, $\vec{x_i}$ is the water saturation, and when simulating the standard black oil model, $\vec{x_i}$ may present the water saturation and the gas saturation. We assume that the well unknowns are denoted by $\vec{w}$, whose dimension equals to the number of wells in the reservoir. Let us define the pressure vector p as follows:

$$p = \begin{pmatrix} p_{o,1} \\ p_{o,2} \\ \cdots \\ p_{o,n} \end{pmatrix}, \tag{3}$$

and the global unknown vector x as follows:

$$x = \begin{pmatrix} p_{o,1} \\ p_{o,2} \\ \cdots \\ p_{o,n} \\ \vec{x_1} \\ \vec{x_2} \\ \cdots \\ \vec{x_n} \\ \vec{w} \end{pmatrix}. \tag{4}$$

Then the global restriction operator is defined as

$$\Pi_r x = p. \tag{5}$$

The global restriction operator Π_r is applied only to the fluid flow variables (not to the well variables). A prolongation operator Π_p is defined as

$$\Pi_p p = \begin{pmatrix} p_{o,1} \\ p_{o,2} \\ \cdots \\ p_{o,n} \\ \vec{0} \\ \vec{0} \\ \cdots \\ \vec{0} \\ \vec{0} \end{pmatrix}. \tag{6}$$

Here $\Pi_p P$ has the same dimension as x.

The matrix A derived from the Newton or Newton-Raphson methods for the black oil model can be written as equation (7) if a proper ordering technique is applied:

$$\widetilde{A} = \begin{pmatrix} A_{pp} & A_{ps} & A_{pw} \\ A_{sp} & A_{ss} & A_{sw} \\ A_{wp} & A_{ws} & A_{ww} \end{pmatrix}, \tag{7}$$

where A_{pp} is the matrix corresponding to the pressure unknowns, A_{ss} is the matrix corresponding to the saturation unknowns, $A_{\omega\omega}$ is the matrix corresponding to the well part, and other matrices are coupled items. From now on, we assume that A has the same structure as $\widetilde{A}$.

Consider the linear system $My = b$, where M is an arbitrary non-singular square matrix and b is the right-hand side. If M is a positive-definite square matrix, then we define the notation $AMG(M)^{-1}b$ to represent the solution y. We should mention that the accuracy for y depends on the termination criteria, such as a tolerance and the number of maximal iterations. For example, when the AMG method is employed in the classical CPR preconditioner, a regular setting for the AMG method is that the number of maximal iterations is fixed at one and then the accuracy is relatively low.

Next, considering the linear system $My = b$, if it is solved by the ILU methods, we use the notation $ILU(M)^{-1}b$ to represent the solution y. In this paper, the ILU methods we use are the ILU(k) and ILUT(p, tol) methods. Finally, if it is solved by the Restricted Additive Schwarz method, then we utilize the notation $R(M)^{-1}b$ to represent solution y.

Multi-Stage Preconditioners

For reservoir simulations, the linear system $Ax = b$ is usually solved by the Krylov subspace solvers, as mentioned above. Basic operations of a Krylov subspace solver include matrix-vector multiplication and vector operations, which have been developed in our platform. During the solution process, a preconditioning system must be solved, which has the same form as $Ax = b$ and can be written as

$$Mx = f.$$

For sequential problems, the ILU methods are general-purpose, which are suitable for all kinds of non-singular matrices. When calculating on parallel computers, the domain decomposition methods are commonly used, one of which is the Restricted Additive Schwarz method [26]. It is implemented in our parallel platform. In each processor, a subproblem is solved by ILU(k) or ILUT(p,tol).

For completeness, the classical CPR preconditioner developed by Wallis et al. is introduced here. Let r be the residual: $r = f - Ax$. Its algorithm is described in Algorithm 1.

Algorithm 1—The classical CPR preconditioner

1: $x = \Pi_p AMG(A_{pp})^{-1}\Pi_r f.$
2: $r = f - Ax$
3: $x = x + ILU(A)^{-1}r.$

Algorithm 2—The CPR preconditioner

1: $x = \Pi_p AMG(A_{pp})^{-1}\Pi_r f.$
2: $r = f - Ax$
3: $x = x + R(A)^{-1}r.$

The preconditioner is a two-stage preconditioner. It solves the pressure matrix A_{pp} first. The solution from the pressure matrix is then used an an approximate solution, and, finally, it solves the whole system using the ILU methods. In our simulator, the algorithm is replaced by Algorithm 2, where the ILU methods are replaced by the RAS method, which is a better choice than the ILU methods when running on parallel systems. From now on, when we say the classical CPR or CPR method, we mean Algorithm 2. We should mention that when using one CPU or one MPI processor, the RAS method is equivalent to the ILU methods.

A variant of the classical CPR preconditioner is described in Algorithm 3, which is noted as a CPR-FP preconditioner in this paper. The only difference is the order of solutions. This preconditioner solves the whole system first, and then solves the pressure matrix. Again, the ILU methods are replaced by the RAS method.

Algorithms

The CPR preconditioner and the CPR-FP preconditioner have been implemented in our platform. According to our experience, the CPR-FP preconditioner has a better convergence than the CPR preconditioner, and it is also more stable. Inspired by this phenomenon, our first CPR-like preconditioner, the CPR-FPF preconditioner, is proposed, which is described in Algorithm 4.

Algorithm 3—The CPR-FP Preconditioner

1: $x = R(A)^{-1} f.$
2: $x = x + \Pi_p AMG(A_{pp})^{-1} \Pi_r r.$

Algorithm 4—The CPR-FPF Preconditioner

1: $x = R(A)^{-1} f.$
2: $x = x + \Pi_p AMG(A_{pp})^{-1} \Pi_r r.$
3: $x = x + R(A)^{-1} r.$

This preconditioner is a three-stage preconditioner, which solves the entire problem first, then the pressure matrix, and the entire problem in the end. When solving on parallel computers, a subdomain problem is solved by the ILU methods on each processor. Details of the RAS method can be read in [28]. The size of each subdomain problem is controlled by the size of the subdomain and the overlap. For parallel computing, if the number of processors is small, such as 32 processors, an overlap of unity is enough. However, when we have hundreds of processors, the overlap should be larger. The accuracy of the solution to each subdomain problem is determined by the solver used to solve the problem. In our platform, the ILU(k) and ILUT(p, tol) methods are used. The default level of ILU(k) is one, and the default tolerance for ILUT(p, tol) is 1e-3. The p from ILUT(p, tol) depends on the subdomain problem.

The algebraic multigrid (AMG) method is applied to solve the pressure matrix A_{pp}. Only one iteration V-cycle AMG method is employed. Based on our experience, one iteration is enough for most problems. However, in our platform, the number of AMG iterations are user-configurable. Other parameters for AMG are also user-configurable, such as the number of levels, the coarsening strategy, the interpolation operators and types of smoothers.

The second one is called a CPR-PFP preconditioner. It solves a pressure subproblem twice and the entire problem once. Its algorithm is proposed in Algorithm 5. This preconditioner is also a three-stage preconditioner. Its settings are the same as those of the CPR-FPF preconditioner. Since one AMG iteration has much more calculations than one RAS iteration, this preconditioner is efficient but slightly slower than the CPR-FPF preconditioner.

Algorithm 5—The CPR-PFP Preconditioner

1: $x = \Pi_p AMG(A_{pp})^{-1}\Pi_r f.$
2: $x = x + R(A)^{-1}r.$
3: $x = x + \Pi_p AMG(A_{pp})^{-1}\Pi_r r.$

The third one is a CPR-FFPF preconditioner, which applies the RAS method twice first and then solves an AMG problem once, and in the end, the RAS method is used again to accelerate the convergence. Its algorithm is shown in Algorithm 6. It is a four-stage preconditioner. Based on our experiments, this preconditioner is efficient and more stable than the CPR-FPF preconditioner. This preconditioner is suitable for extremely difficult black oil simulations and polymer flooding simulations.

Algorithm 6—The CPR-FFPF Preconditioner

1: $x = R(A)^{-1}f.$
2: $x = x + R(A)^{-1}r.$
3: $x = x + \Pi_p AMG(A_{pp})^{-1}\Pi_r r.$
4: $x = x + R(A)^{-1}r.$

Matrix Processing Techniques

It is well-known that proper matrix reordering techniques improve the efficiency of linear solvers and preconditioners. Some techniques reduce the bandwidth of a given matrix, such as the reverse Cuthill-MaKee reordering method. Some techniques reduce fill-ins of the ILU methods, such as the minimum degree reordering method. Other techniques reduce the condition number of a given matrix. Proper reordering techniques are important to the development of linear solvers and preconditioners.

For reservoir simulation, upstream schemes are widely used. An upstream is determined by the potential of a phase. Kwok and Tchelepi developed a potential-based reordering strategy and a reduced Newton algorithm was designed [47]. In this paper, the potential-based reordering strategy is applied to reorder the pressure unknowns and the saturation unknowns. Since a global reordering is extremely difficult, the reordering strategy is applied locally on each processor. On each processor, the potential of each grid block is sorted decreasingly. Then all unknowns in each grid block have the same order as the potential. In this case, the numbering of all unknowns has been changed and this process must be repeated at each Newton iteration.

In order to weaken the coupling between the pressure unknowns and saturation unknowns, the alternative block factorization (ABF) strategy [18] is applied. The decoupling strategy can be described as in (8):

$$D^{-1}Ax = D^{-1}b, \tag{8}$$

where D is a block diagonal matrix if a proper numbering technique is applied. In this case, the D has the form as (9):

$$D = \mathrm{Diag}(\mathrm{D}_1, \mathrm{D}_2, \cdots, \mathrm{D}_\mathrm{n}, \mathrm{I}_\mathrm{w}), \tag{9}$$

where D_i is an $m \times m$ ($2 \leq m \leq 3$) matrix for the two-phase model or the black oil model. The resulting linear system is also denoted by $Ax = b$. The details of the ABF strategy can be found in [18].

When solving a linear system, the matrix decoupling technique applies first, and then the reordering technique applies to the decoupled system. In the end, the linear system is solved by Krylov subspace

solvers with the preconditioners developed in this paper. The linear system is equivalent to the original linear system. In our simulator, these two techniques are both optional.

Numerical Experiments

Numerical experiments are performed in this section for preconditioners, which are implemented based on PRSI, a general purpose parallel platform for reservoir simulators developed by our group. The PRSI platform is written in C and MPI (Message Passing Interface), which provides a grid management module, a data management module, a linear solvers module, a preconditioners module, a distributed matrices and vectors management module, a visualization module, a key words parsing module and a well modeling module. The grids are structured Cartesian grids, which support the finite difference methods and finite volume methods. The finite element methods may be supported in the future. The load balancing methods for grid partitioning are the ParMETIS [64], Zoltan [50], and Hilbert space-filling curve methods [51]. Other partitioning methods are also supported, such as the Hamilton path method [52, 53] and the Morton space-filling curve method. The Krylov subspace solvers and AMG solvers are provided, including GMRES, BICGSTAB [13] and the classical AMG solver [48, 49] from HYPRE. General purpose preconditioners, including ILU, block-wise ILU, the RAS method [28] and AMG [48, 49], and special preconditioners, including CPR [16, 17, 30, 65] and the preconditioners introduced in this paper, are implemented. For the well modeling module, Peaceman models [46, 66] are implemented.

All numerical experiments are tested on the cluster Parallel from Westgrid, University of Calgary, and an IBM Blue Gene/Q supercomputer from IBM. The Parallel cluster has 7,056 CPU cores, which has 528 12-core standard nodes and 60 special 12-core nodes with 3 NVIDIA Tesla GPUs. The 12 cores associated with one compute node share 24 GB of RAM. Parallel uses an InfiniBand 4X QDR (Quad Data Rate) 40 Gbit/s switched fabric, with a two to one blocking factor.

The IBM Blue Gene/Q we use has two racks, and each rack has two midplanes. Each midplane has 16 computer nodes and each node has 32 computer cards (64-bit PowerPC A2 processor). Each computer card has 17 cores, one of which is for the operation system and the other 16 cores for computation. The 64-bit PowerPC A2 processor cores are 4-way simultaneously multi-threaded, and run at 1.6 GHz. Each core has a SIMD Quad-vector double precision floating point unit (IBM QPX). The IBM Blue Gene/Q has 32,768 CPU cores.

Before we discuss our experiments, we should mention that the running time of each case in the first two parts is not reported. The reason is that the cluster we use is a production machine, which is open to hundreds of researchers. The cluster is always fully occupied. Each computer node is shared by several programs and these programs compete the bandwidth, too. The running time of each program is highly affected by other programs, which makes it hard to obtain accurate running time.

An Oil-water Model

The oil-water model is a simplified version of the black oil model, which assumes that there are the water and oil components and the water and oil phases. There is no mass transfer between the two phases. The SPE Tenth Comparative Project (SPE10) [57] is applied to test our preconditioners. The model has a sufficiently fine grid to make use of classical pseudoisation methods almost impossible. At the fine geological model scale, the model is described on a regular Cartesian grid. The model dimensions are 1, 200 $\times$ 2, 200 $\times$ 170 (ft). The top 70 ft (35 layers) represents the Tarbert formation, and the bottom 100 ft (50 layers) represents Upper Ness. The fine scale cell size is 20 $\times$ 10 $\times$ 2 (ft) [57]. The grid size of the model is 60 $\times$ 220 $\times$ 85 blocks (1.122 $\times$ 10^6 blocks). It has five wells, one injection well and four production wells. The oil-water model in this case has around 2.244 millions of unknowns. The details can be found in [57].

The model is a highly heterogeneous model, which makes the linear systems extremely difficult to solve. The setting makes this project a famous benchmark for testing the design of linear solvers and

preconditioners. A lot of papers have been published to solve this type of problems, and details can be read in [16, 17, 30, 65].

Example 1 *This example tests the SPE10 project. The purpose is to test the efficiency of matrix preprocessing techniques and two sets of experiments are carried. The experiments are run on the Parallel cluster from University of Calgary. The first set uses the matrix decoupling strategy ABF and the local potential reordering strategy. The second set does not use the matrix decoupling strategy ABF and the local potential reordering strategy. The inexact Newton method is applied to both sets and the termination tolerance is* 10^{-2}. *The number of maximal Newton iterations is 20. The linear solver is GMRES(100) and the number of maximal inner iterations is 200. The overlap for the RAS method is fixed at 1 and ILUT(p,1e-3) is used as the solver for each subdomain problem. The simulation period is 2,000 days and the maximal time step is 100 days. Two computation nodes are used, each node has eight processors, and the total 16 MPI processors are employed to run the simulations. Summaries of the two sets are shown in Tables 1 and 2, respectively.*

Table 1—Summary of the first set, Example 1

Preconditioner	# Time steps	# Newton	# Solver
CPR	55	363	23053
CPR-PFP	55	364	19503
CPR-FP	55	364	23086
CPR-FPF	55	370	16785
CPR-FFPF	56	379	14321

Table 2—Summary of the second set, Example 1

Preconditioner	# Time steps	# Newton	# Solver
CPR	74	580	71039
CPR-PFP	63	455	45212
CPR-FP	88	693	81143
CPR-FPF	NA	NA	NA
CPR-FFPF	NA	NA	NA

Tables 1 and 2 collect the number of time steps, the number of Newton iterations and the number of linear solver iterations. For GMRES(m), the number of linear solver iterations is the number of inner iterations. Five preconditioners are employed. From Table 1, we can see that the number of time steps for each preconditioner is similar and the number of Newton iterations does not vary too much. When the classical CPR preconditioner is applied, the number of Newton iterations is 363. The number of Newton iterations for the CPR-PFP preconditioner and the CPR-FP preconditioner is 364. The number of Newton iterations for the CPR-FPF preconditioner is 370, which is slightly more than the CPR, CPR-PFP and CPR-FP preconditioners. The CPR-FFPF uses 379 Newton iterations, and it uses more than the other four preconditioners. However, the number of linear solver iterations varies from 14,321 to 23,086. The classical CPR preconditioner and the variant CPR-FP preconditioner have the worst performance for this example, which use around 23,000 iterations. The CPR-FFPF preconditioner has the best performance, which uses only 14,321 iterations, followed by the CPR-FPF preconditioner and the CPR-PFP preconditioner.

From Table 2, we can see that the CPR-FFPF preconditioner and the CPR-FPF preconditioner failed the tests. In this group, the CPR-PFP preconditioner has the best performance, which uses the fewest number of time steps, the fewest number of Newton iterations and the fewest number of linear solver iterations. Moreover, the CPR preconditioner is better than the CPR-FP preconditioner, which means that the pressure unknowns contribute the most error.

Comparing the results from Tables 1 and 2, we can conclude that the matrix decoupling strategy and the matrix reordering strategy are critical to the success of physics-based preconditioners. When these techniques are applied, the CPR-FPF preconditioner and the CPR-FFPF preconditioner have the best effect, and if they are disabled, these two preconditioners failed. When the techniques are applied, the number of linear solver iterations using the classical CPR preconditioner is reduced from 71,039 to 23,053. The number of linear solver iterations using the CPR-FP preconditioner is reduced from 81,143 to 23,086. The matrix techniques also reduce the number of time steps and the number of Newton iterations.

To our experience, the same phenomenon occurs in the standard black oil simulations. In our simulator, these two techniques are enabled by default. From now on, all the experiments below use these two techniques by default.

Example 2 *This example tests the parallel stability of our preconditioners using the SPE10 project. It is run on the Parallel cluster. The inexact Newton method is applied and the termination tolerance is* 10^{-2}. *The number of maximal Newton iterations is 20. The linear solver GMRES(100) is chosen and its number of maximal inner iterations is 200. The overlap for the RAS method is also fixed at 1 and ILUT(p,1e-3) is used as a local solver for each subdomain problem from the RAS method. Again the simulation period is 2,000 days and the maximal time step is 200 days. Each computer node has eight processors, and up to 8 nodes are employed. Up to 64 MPI processors are employed to run the simulations. Summaries of numerical results are shown in Tables 3-7.*

This example tests the stability of the CPR-like preconditioners corresponding to the number of CPU cores. In this paper, the RAS method is used to replace the regular ILU methods. It is well-known that the RAS method has better parallel scalability than the ILU methods; however, the approximation of each subdomain problem to the entire system becomes weaker as the number of processors increases and the subdomain on each processor becomes smaller. As a consequence, the number of linear solver iterations may increase and the performance of reservoir simulators may be reduced.

Table 3—Summary of the CPR preconditioner, Example 2

# Procs	# Time steps	# Newton	# Solver
8	53	357	23462
16	54(2)	396	24937
32	53	356	23524
64	56	386	25547

Table 4—Summary of the CPR-PFP preconditioner, Example 2

# Procs	# Time steps	# Newton	# Solver
8	52	346	19508
16	53	356	19743
32	53	353	20067
64	54	381	21493

Table 5—Summary of the CPR-FP preconditioner, Example 2

# Procs	# Time steps	# Newton	# Solver
8	54	366	25201
16	54	368	22936
32	53	360	23850
64	53	361	26851

Table 6—Summary of the CPR-FPF preconditioner, Example 2

# Procs	# Time steps	# Newton	# Solver
8	54	365	16537
16	53	358	16366
32	53	359	16425
64	54	364	15667

Table 7—Summary of the CPR-FFPF preconditioner, Example 2

# Procs	# Time steps	# Newton	# Solver
8	54	366	14077
16	53	362	13885
32	54	364	14573
64	53	360	12526

Table 3 shows numerical results for the CPR preconditioner. 8, 16, 32 and 64 MPI processors are employed for each run. The number of time steps is stable, which increases from 53 to 56. When using 16 MPI processors, the inexact Newton method failed twice. From this table, we can see that the number of Newton iterations for the 16 MPI processors is the most due to the failure of Newton method. It also has the most number of linear solver iterations. When increasing the number of processors, the number of Newton iterations increases from 357 to 386. The number of linear solver iterations also increases slightly, from 23,462 to 25,547. The data shows that the preconditioner becomes slightly weaker when increasing the number of CPU cores (or MPI processors).

Table 4 shows numerical results for the CPR-PFP preconditioner. This table shows that the number of time steps increases, which varies from 52 to 54. The number of Newton iterations also increases, from 346 to 381. However, when increasing the number of MPI processors from 8 to 32, the change is small comparing to the 64 MPI processors case. The number of linear solver iterations increases when using

Table 8—Oil PVT

Pressure psia	Rs MSCF/STB	Bo RB/STB	Viscosity cp	Co psia^{-1}	Cvo psia^{-1}
400	0.0165	1.012	3.5057	1.1388e-6	0.0
4000	0.1130	1.01278	2.9972	1.1388e-6	0.0
10000	0.1810	1.155	2.6675	1.1388e-6	0.0

Table 9—Gas PVT

Pressure psia	Bg SCF/STB	Viscosity cp
400	1.96	0.0140
4000	0.84	0.0160
8000	0.59	0.0175
10000	0.42	0.0195

more MPI processors, from 19,508 to 21,493. Comparing with Table 3, this preconditioner is slightly more stable than the CPR preconditioner.

Table 5 shows numerical results for the CPR-FP preconditioner. This table shows different tendency. By increasing the number of processors, the number of time steps and the number of Newton iterations decrease. The number of linear solver iterations also decreases untill 32 MPI processors. When using 64 MPI processors, the number of linear solver iteration is the most.

Table 6 presents results for the CPR-FPF preconditioner, which shows that the number of time steps does not vary too much. The number of Newton iterations increases from 8 MPI processors to 32 MPI processors. When using 64 processors, the number of Newton iterations increases to 364. The number of linear solver iterations is interesting. The 8 processors has the most iterations. By using more processors, the number of linear solver iterations decreases. When using 64 processors, the number of linear solver iterations is the fewest. This table also shows that the CPR-FPF preconditioner is more stable than the CPR preconditioner, the CPR-PFP preconditioner and the CPR-FP preconditioner, and it uses the fewest number of linear solver iterations than others.

Table 7 shows results for the CPR-FFPF preconditioner. We can see that all data is similar to that of the CPR-FPF preconditioner except the number of linear solver iterations. All cases have fewer iterations than the other preconditioners. It is the most effective preconditioner.

From Tables 3 - 7, we can see that the CPR-FFPF is the best preconditioner, followed by the CPR-FPF preconditioner and the CPR-PFP preconditioner. The classical CPR and the CPR-FP have the worst performance. For each preconditioner, it is stable when using different numbers of processors.

We can see that our preconditioners are efficient for models with highly heterogeneous geological properties. Compared with the last example, we can see that the potential reordering and matrix decoupling techniques are very important to the solution of reservoir simulations.

The Standard Black Oil Model

In this section, two examples are studied. The first one has the same grid and geological model as the SPE10 project. This model also has five wells, one injection well and four production wells. This is also an extremely hard problem to be solved. The second case is the refined version of SPE1 [54]. It has two wells. The model is simple and easier to solve. The example is employed to show the effectiveness of our preconditioners to simple models.

The standard SPE10 project is for an oil-water model. Its fluid properties are adopted; see Tables 8 and 9. B_o is the oil formation volume factor, R_s is the solution gas-oil ratio for saturated oil, C_o is the compressibility of undersaturated oil and C_{vo} is the oil viscosity compressibility of undersaturated oil. The water formation volume factor is calculated by

$$B_w = \frac{B_w^0}{1 + C_w(p - p^0)}, \tag{10}$$

where $B_w^0 = 1.01$, $C_\omega = 3 \times 10^{-6}$ psia ^{1} and $p^0 = 6{,}000$ psia. The water viscosity $\mu_\omega = 0.3$cp is constant.

Example 3 *This example tests the efficiency of our preconditioners for the black oil model using the SPE10 simulation model. The experiments are run on the Parallel cluster. The inexact Newton method is applied, the termination tolerance is* 10^{-3} *and the number of maximal Newton iterations is 15. The BICGSTAB linear solver is chosen and the number of maximal inner iterations is 100. The overlap for the RAS method is fixed at 1 and ILUT(p,1e-3) is used for each subdomain problem. The simulation period is 20 days. Each computer node has eight processors, and up to 8 nodes are employed. Summaries of numerical results are shown in Tables 10 - 14.*

The SPE10 problem for the black oil model has 1.122 millions of grid blocks and around 3.366 millions of unknowns. Table 10 is the summary of the CPR preconditioner. The number of time steps for each case is the same, which is 69. The number of Newton iterations increases from 212 to 230. The number of linear solver iterations increases when using more processors, from 1,491 to 1,919. The average number of linear solver iterations is less than 10, which means that the CPR preconditioner is effective in the case. The numerical results for the CPR-PFP preconditioner in Table 11 shows the same trend. However, the CPR-PFP preconditioner uses fewer Newton iterations and linear solver iterations. Its average number of linear solver iterations is also fewer than 10. The results show that the CPR-PFP is more effective than the classical CPR preconditioner.

Table 10—Summary of the CPR preconditioner, Example 3

# Procs	# Time steps	# Newton	# Solver
8	69	212	1491
16	69	230	1533
32	69	230	1567
64	69	230	1919

Table 11—Summary of the CPR-PFP preconditioner, Example 3

# Procs	# Time steps	# Newton	# Solver
8	69	206	937
16	69	214	1026
32	69	213	1117
64	69	223	1704

Table 12—Summary of the CPR-FP preconditioner, Example 3

# Procs	# Time steps	# Newton	# Solver
8	69	228	1110
16	69	224	1245
32	69	229	1313
64	69	234	1547

Table 13—Summary of the CPR-FPF preconditioner, Example 3

# Procs	# Time steps	# Newton	# Solver
8	69	203	1101
16	69	207	1165
32	69	210	1230
64	69	218	1580

Table 14—Summary of the CPR-FFPF preconditioner, Example 3

# Procs	# Time steps	# Newton	# Solver
8	69	201	999
16	69	205	1164
32	69	207	1166
64	69	209	1494

Table 12 shows the numerical results for the CPR-FP preconditioner, from which we can see that the number of time steps is the same, the number of Newton iterations is similar, and the number of linear solver iterations increases from 1,110 to 1,547 when increasing MPI processors. The average number of linear solver iterations per Newton iteration is fewer than 8. The results mean that the CPR-FP preconditioner is better than the classical CPR and CPR-PFP preconditioners. The CPR-FPF preconditioner shows similar results in Table 13 with the CPR-FP preconditioner, which requires fewer Newton iterations and linear solver iterations except the 64 processors case.

Table 14 is a summary for the CPR-FFPF preconditioner. The data shows that it uses the fewest Newton iterations and linear solver iterations. The number of Newton iterations just increases slightly, from 201 to 209. The number of linear solver iterations increase from 999 to 1,494. The average number of linear solver iterations is fewer than 8. We can see that it is the most effective preconditioner here.

Example 4 *This example tests the efficiency of our preconditioners for simple models. The experiments are run on the Parallel cluster. The refined SPE1 project is applied. The model has 2.5 millions of grid blocks and two wells. The model has around 7.5 millions of unknowns. The inexact Newton method is applied and the termination tolerance is* 10^{-3}. *The experiments terminate in 20 time steps. BICGSTAB is the linear solver. The number of maximal inner iterations is 200. The overlap for the RAS method is fixed at 1 and ILUT(p,1e-3) is used for each subdomain problem. Each computer node has eight processors, and up to 8 nodes are employed. Summaries of numerical results are shown in Tables 15 - 19.*

The classical CPR preconditioner and the CPR-PFP preconditioner have the same data, which is shown in Tables 15 and 16. The data does not change with different numbers of processors. The number of time steps is fixed at 20, and 40 Newton iterations are required. Each time step requires two Newton iterations. A total of 54 linear solver iterations is observed. The average number of linear solver iterations is fewer than two. It means that the two preconditioners are effective and stable. When the CPR-FP preconditioner is used, the number of Newton iterations and the number of linear solver iterations are slightly more, which are 42 and 59, respectively.

The CPR-FPF preconditioner and the CPR-FFPF preconditioner have the same data, which is shown in Tables 18 and 19. The data does not change when changing the number of MPI processors. For these two preconditioers, the linear solver converges in one iteration. We can see that these two preconditioners are extremely effective for the black oil model with simple geological data.

Table 15—Summary of the CPR preconditioner, Example 4

# Procs	# Time steps	# Newton	# Solver
8	20	40	54
16	20	40	54
32	20	40	54
64	20	40	54

Table 16—Summary of the CPR-PFP preconditioner, Example 4

# Procs	# Time steps	# Newton	# Solver
8	20	40	54
16	20	40	54
32	20	40	54
64	20	40	54

Table 17—Summary of the CPR-FP preconditioner, Example 4

# Procs	# Time steps	# Newton	# Solver
8	20	42	59
16	20	42	59
32	20	42	59
64	20	42	59

Table 18—Summary of the CPR-FPF preconditioner, Example 4

# Procs	# Time steps	# Newton	# Solver
8	20	39	39
16	20	39	39
32	20	39	39
64	20	39	39

Table 19—Summary of the CPR-FFPF preconditioner, Example 4

# Procs	# Time steps	# Newton	# Solver
8	20	39	39
16	20	39	39
32	20	39	39
64	20	39	39

This example shows that our preconditioners are effective and stable for simple black oil problems with millions of grid blocks. It also demonstrates that the CPR-FPF and CPR-FFPF preconditioners are more effective than the other three preconditioners.

Scalability

This subsection tests the scalability of our parallel preconditioners. Three examples are presented. The first one tests a small model with 2.5 millions of grid blocks and uses up to 256 CPU cores. The second example is a simple oil-water model with 100 millions of grid blocks. The third example is the refined SPE1 project, which tests a large-scale black oil model with 100 millions of grid blocks and uses up to 3,072 CPU cores.

The speedup s is defined as

$$s = t_b/t_p,$$

where t_b is the running time of the base case and t_p is the running time of other cases. A typical choice for the base case is one processor case. When s is m, it means that the current case is m times faster than the base case. An ideal condition is that when doubling the number of processors, the speedup doubles. The efficiency of parallel computing is defined as

$$e = t_b * N_b/(t_p * N_p) = s * N_b/N_p,$$

where N_b is the number of processors (CPUs or CPU cores) for the base case and N_p is the number of processors (CPUs or CPU cores) for other cases. Again, a typical choice for N_b is one processor. The ideal condition is that the efficiency is one.

Example 5 *The example tests the scalability of each preconditioner implemented in our black oil simulator. The problem is also the SPE1 project with 2.5 millions of grid blocks and 7.5 millions of unknowns. The inexact Newton method is applied and the termination tolerance is* 10^{-3}. *The linear solver is BICGSTAB. The number of maximal inner iterations is 100. The overlap for the RAS method is fixed at 1 and ILUT is used for each subdomain problem. Again, the simulation period is fixed at 3 time steps. The experiments are performed on IBM Blue Gene/Q, and up to 256 MPI processors are employed. The 16 MPI processors case is used as the base when calculating speedup. Summaries of numerical results are shown in Tables 20 - 24 and Figs. 1 - 5.*

Table 20—Summary of the CPR preconditioner, Example 5

# Procs	# Newton	# Solver	Time (s)	Speedup
16	6	8	1551	1
32	6	8	782.5	1.98
64	6	8	389.5	3.98
128	6	8	198.2	7.82
256	6	8	117.3	13.22

Table 21—Summary of the CPR-PFP preconditioner, Example 5

# Procs	# Newton	# Solver	Time (s)	Speedup
16	6	8	1542.8	1
32	6	8	778.7	1.98
64	6	8	386.2	3.99
128	6	8	196.3	7.86
256	6	8	119.5	12.91

Table 22—Summary of the CPR-FP preconditioner, Example 5

# Procs	# Newton	# Solver	Time (s)	Speedup
16	6	8	1571	1
32	6	8	790	1.99
64	6	8	392	4.01
128	6	8	200.7	7.83
256	6	8	117.7	13.35

Table 23—Summary of the CPR-FPF preconditioner, Example 5

# Procs	# Newton	# Solver	Time (s)	Speedup
16	6	6	1456.3	1
32	6	6	735.5	1.98
64	6	6	364.5	4.0
128	6	6	185.5	7.85
256	6	6	111	13.12

Table 24—Summary of the CPR-FFPF preconditioner, Example 5

# Procs	# Newton	# Solver	Time (s)	Speedup
16	6	6	1506.5	1
32	6	6	740.2	2.04
64	6	6	366.6	4.11
128	6	6	186.6	8.07
256	6	6	102.8	14.65

The results for the CPR preconditioner are in Table 20, where the 16 processors case is the base case. When doubling processors, the speedup is 1.98. The ideal speedup for 32 MPI processors is 2.0, which means our preconditioner and our simulator almost have ideal scalability. When using 64 processors, the speedup is 3.98 while the ideal value is 4.0. Again, the scalability is nearly ideal. When using 128 processors, our speedup is 7.82. The scalability is linear. When increasing the processors to 256, each processor has fewer than 10,000 grid blocks, the communication between processors becomes dominant, and we can see that the speedup is 13.22, which is still good, but the efficiency is low, which is only 0.826 (= 13.22/16). From Table 20, we can see that the RAS method has good scalability and good parallel efficiency. The experiments also demonstrate that the AMG method has good scalability. The CPR preconditioner in our simulator is scalable.

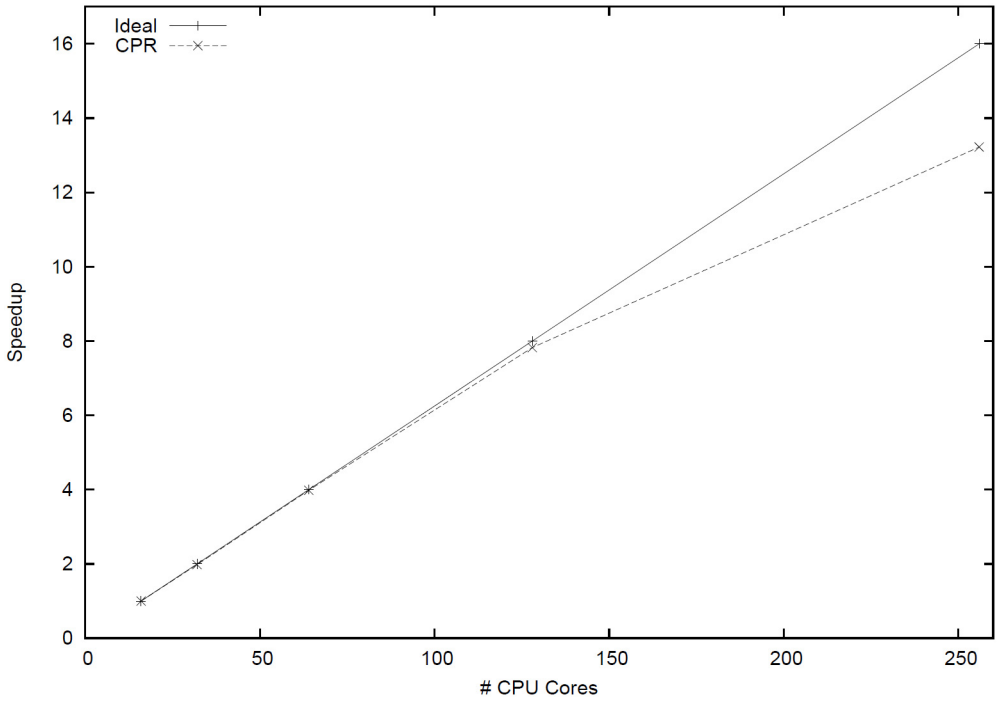

Figure 1—Speedup of CPR, Example 5

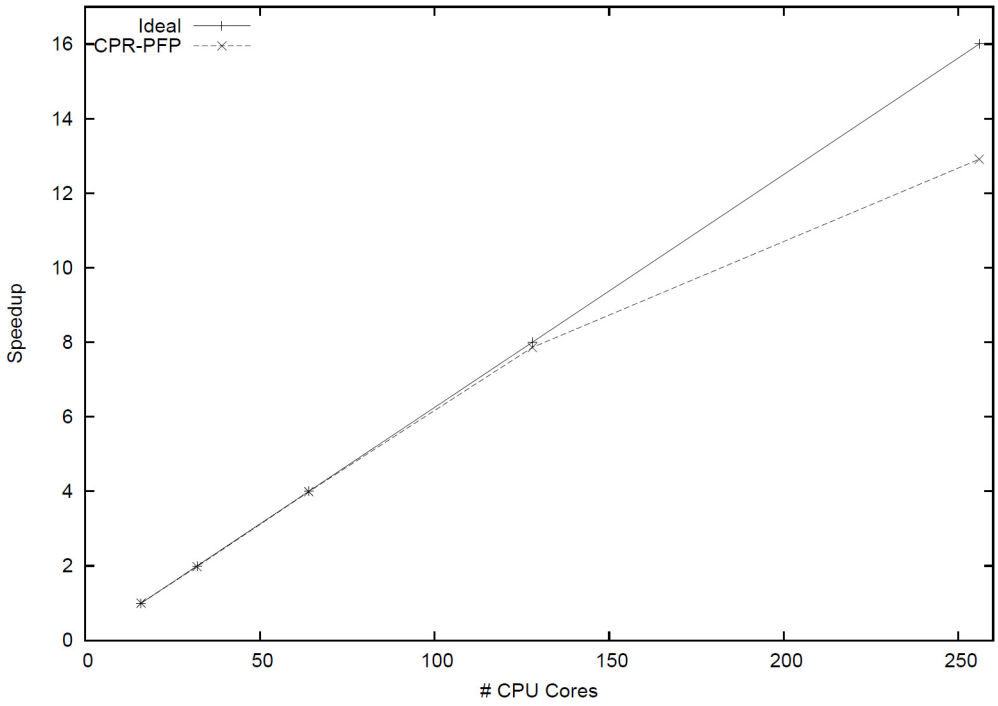

Figure 2- Speedup of CPR-PFP, Example 5

The results for the CPR-PFP preconditioner are in Table 21, which also shows good scalability. When doubling the processors, the speedup is 1.98. When using 64 processors, the speedup is 3.99, which is almost perfect. When increasing the number of processors to 128, the speedup is 7.86 and the efficiency is 0.983. The efficiency is perfect in practice. When using 256 processors, each processor has only 0.39%

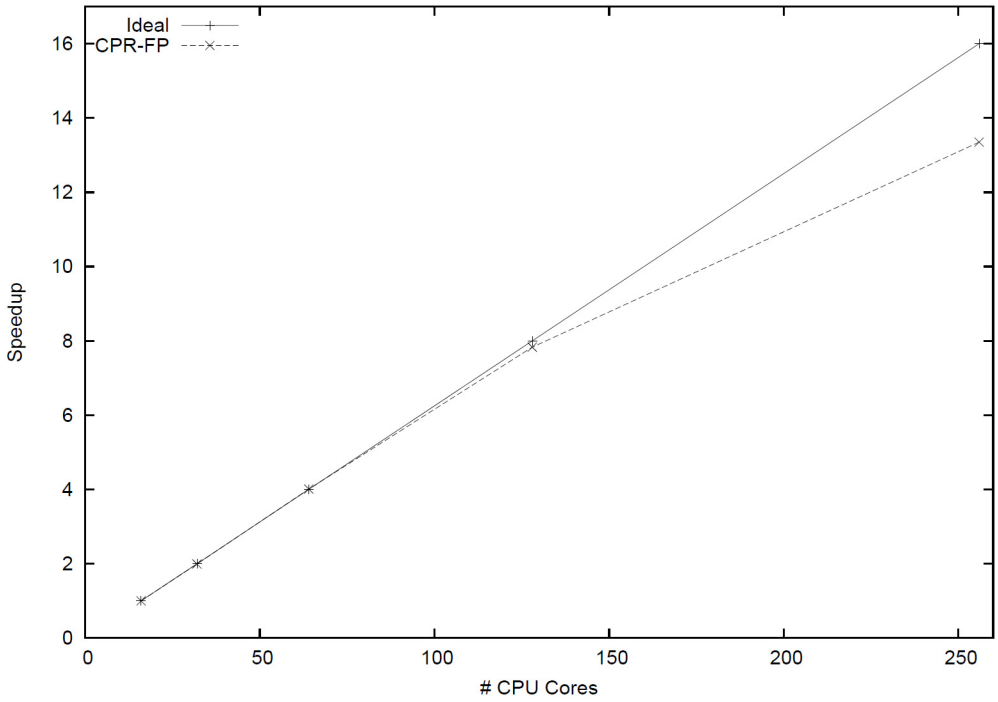

Figure 3—Speedup of CPR-FP, Example 5

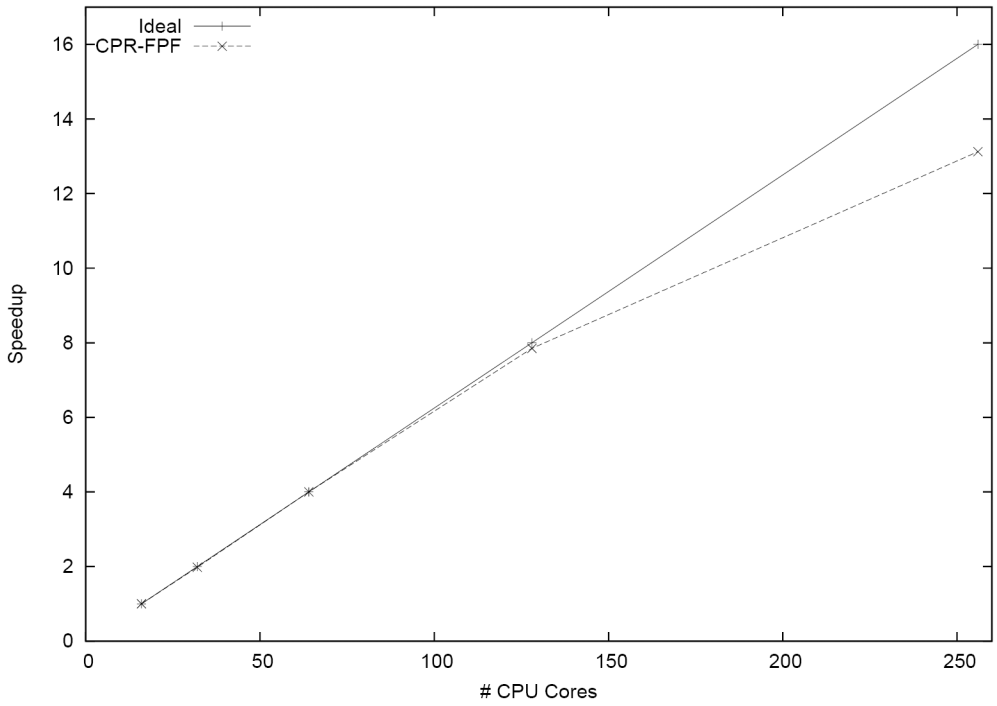

Figure 4—Speedup of CPR-FPF, Example 5

of the total grid blocks, and a speedup of 12.91 is obtained. We can conclude that the CPR-PFP is also scalable.

The results for the CPR-FP and CPR-FPF preconditioners are similar, which are shown in Tables 22 and 23. When using up to 128 processors, the efficiency is almost ideal. When using 256 processors, the efficiency becomes relative low. The results prove that these two preconditioners have good scalability.

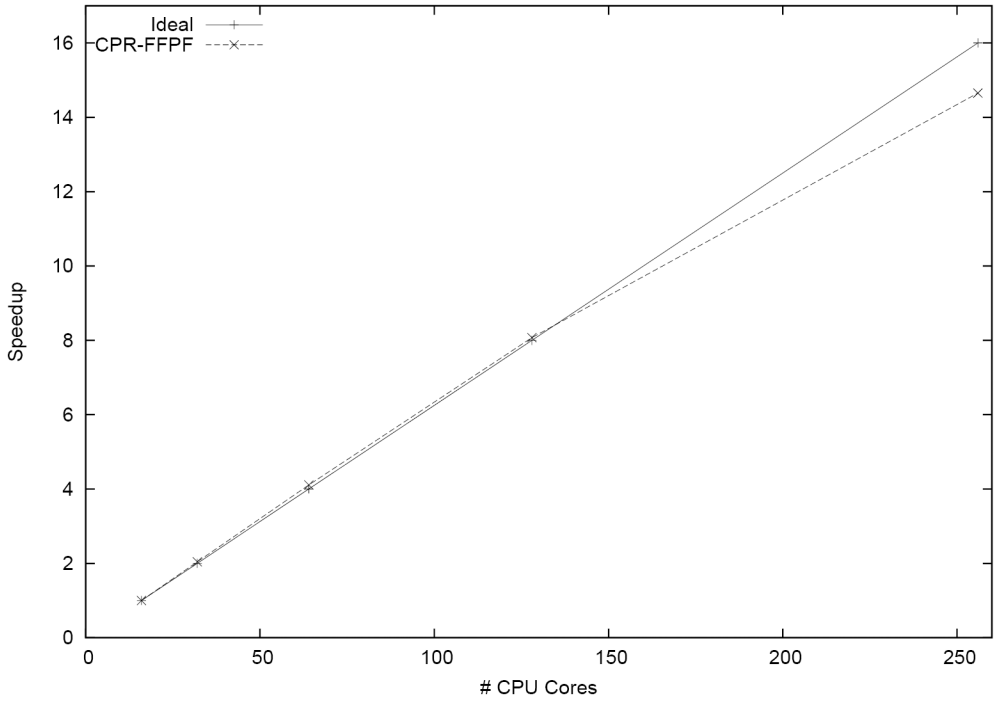

Figure 5—Speedup of CPR-FFPF, Example 5

Table 25—Summary of the CPR-FPF preconditioner, Example 6

# Procs	# Newton	# Solver	Time (s)	Speedup	Efficiency
32	42	76	3755.4	1	100%
64	42	80	1925.8	1.95	97.5%
96	42	80	1183.2	3.17	105.6%
128	44	76	933.5	4.02	100.1%
160	42	76	731.4	5.13	102.6%

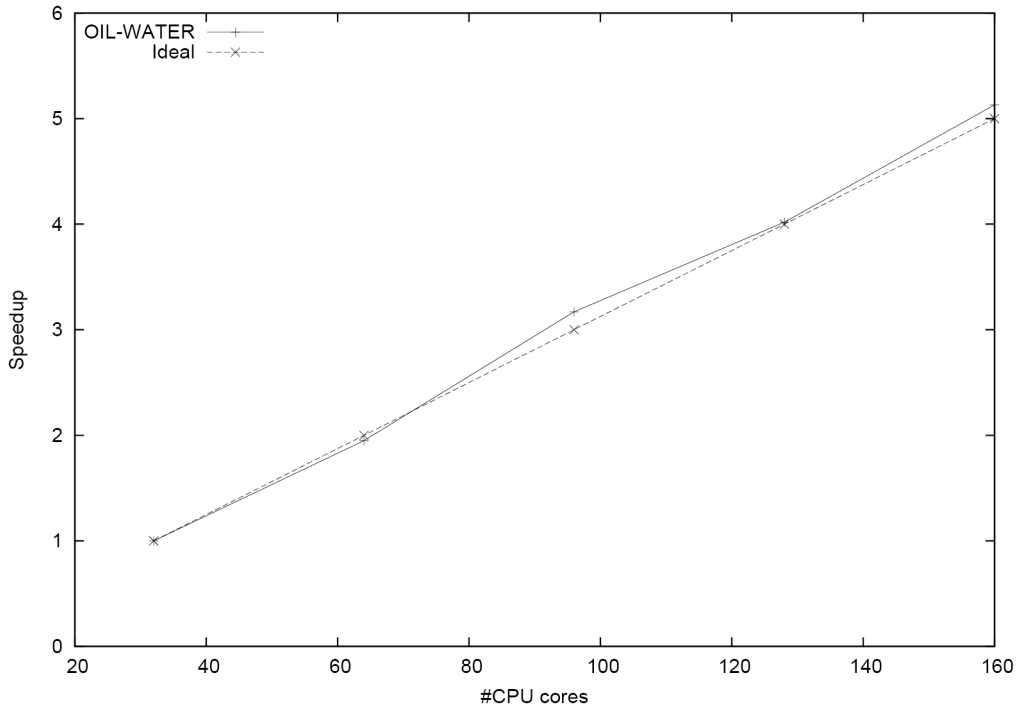

Figure 6—Speedup of CPR-FPF, Example 6

Table 26—Summaries of scalability for the SPE10 project, Example 7

# procs	# Newton	# Linear	Time (s)	Speedup	Efficiency
4	476	7382	8678	1.	100%
8	478	7719	5201	1.67	83.5%
16	460	7488	2272	3.82	95.5%
32	468	8308	1279	6.78	84.8%
64	480	8914	693	12.52	78.3%
128	473	10772	424	20.47	64.0%
512(128 × 4)	472	10773	279	31.10	24.3%

Table 24 is for the CPR-FFPF preconditioner. This table shows that super linear scalability is obtained using up to 128 processors. When using 32 processors, the speedup is 2.04 and the efficiency is 1.02, while the speedup and the efficiency of the ideal condition is 2 and 1, respectively. When using 64 processors, the speedup and efficiency is better than those of the ideal condition. When using 128 processors, the speedup is 8.07, which shows super linear scalability. The speedup for 256 processors is slightly low, only 14.65, and its efficiency is 0.916. This preconditioner shows better scalability than other preconditioners.

From Figs. 1 - 5, we can see that all preconditioners show good scalability and they have similar scalability. The scalability using up 128 CPU cores is almost ideal. However, when we use 256 CPU cores, the efficiency of these preconditioners are slightly lower. The reason is that the grid blocks in each processor is less than 10,000 and the communication is becoming dominant. We can conclude that these preconditioners are scalable. The RAS method that is used to replace the ILU methods is scalable and it is a good choice for parallel preconditioners. The AMG method for solving the pressure matrix is also scalable.

Example 6 *The example tests the scalability of the preconditioner CPR-FPF. The problem is a simple oil-water model with 100 millions of grid blocks and 200 millions of unknowns. The inexact Newton method is applied and the termination tolerance is* 10^{-3}. *The linear solver is BICGSTAB. The overlap for the RAS method is fixed at 1 and ILUT is used for each subdomain problem. The simulation period is fixed at 20 time steps. The experiments are performed on the IBM Blue Gene/Q. This case uses up to 160 nodes and the 32 node case is used as the base case, whose speedup is assumed to be one. Summaries of numerical results are shown in Table 25 and speedup is shown in Fig. 6.*

From Example 5, we know that these preconditioners have similar scalability, and in this example, only the CPR-FPF is applied to test the large-scale simulation. From Table 25, we can see the inexact Newton method is effective and stable. Each time step requires around two Newton iterations. In each Newton iteration, less than two linear iterations are required. It means the CPR-FPF preconditioner is effective. In this example, the 32 node case is assumed as the base case and its speedup is one. When using 64 nodes, the speedup is 1.95 and we have an efficiency of 97.5% while the ideal efficiency is 100%. When 96 nodes are used, the speedup is 3.17 and the efficiency is 105.6%, which shows super-linear scalability. When 128 nodes are used, the speedup is 4.02. When 160 nodes are employed, the speedup is 5.13, which means it is 5.13 times faster than the case of 32 nodes. Fig. 6 shows the parallel computational methods developed in this paper have excellent scalability and these methods are efficient for large-scale simulations.

Example 7 *This example studies the SPE10 project. The stopping criterion for the inexact Newton method is 1e-2 and the maximal Newton iterations are 20. The GMRES(100) method and the CPR-FPF preconditioner are applied. The maximal number of inner iterations the GMRES(100) is 200. The Quasi-IMPES strategies are applied. The overlap for RAS method is 1. The simulation period is 2,000 days and the maximal time step is 100 days. The numerical summaries are shown by Table 26 and the speedup is presented by the Figure 7. This example is run on the Parallel cluster, and only one core of*

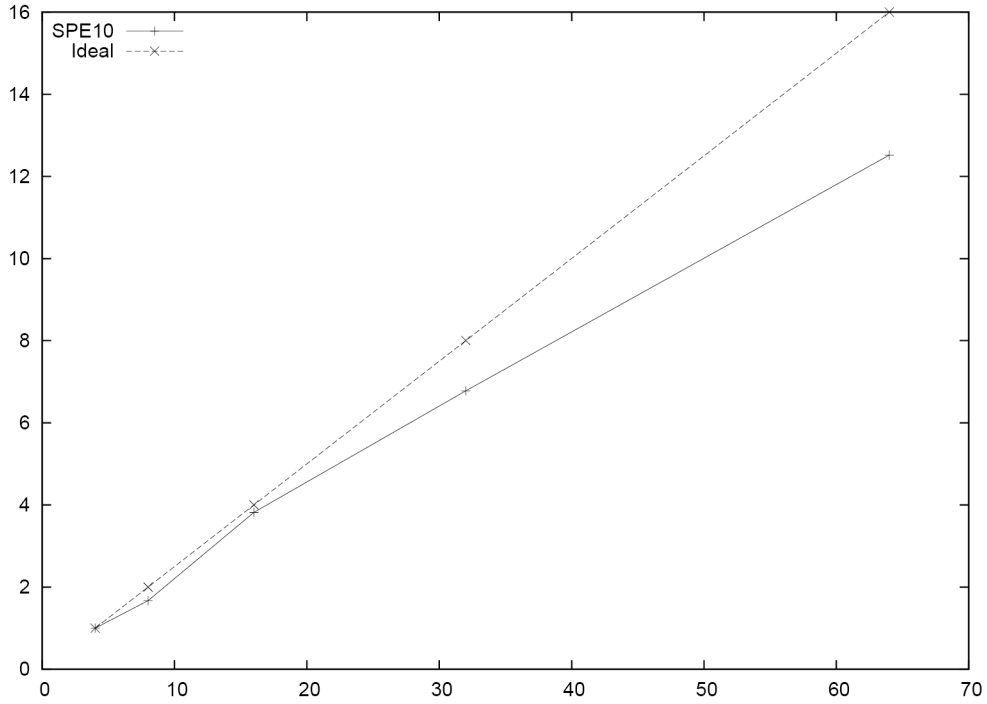

Figure 7—Speedup of the SPE10 project

each computation node is employed except the last simulation, which uses four cores of each computation node.

This case has 1.122×10^6 grid blocks. In practice, 64 CPU cores or less are enough. Here, up to 512 CPU cores are employed. The reason we use 128 and 512 CPU cores is that we would like to see how fast our simulator is. From Table 26, we can see that the inexact Newton method is stable, which consumes around 470 iterations for all simulations. The linear iterations increase when we use more and more CPU cores. However, the average linear iterations for each Newton iteration are less than 23, which is also efficient. The running time decreases when using more CPU cores, from 8,678 seconds to 279 seconds. To our knowledge, the running time of the 279 seconds (4.65 minutes) is the fastest in the world. The simulation with four CPU cores is the base case, whose speedup is assumed to be one. The ideal efficiency is 100%. When we use 64 or less CPU cores, the scalability of our methods are good. For our simulator, the efficiency is 95.5% when using 16 CPU cores and it is 84.8% when using 32 CPU cores. When using 64 CPU cores, the efficiency is 78.3%. And when 128 CPU cores are employed, the efficiency is 64%. When using 512 CPU cores, the efficiency is 24.3% only. The reason is that each CPU core has around 2,200 grid blocks and the communication is dominant compared to the computations. The speedup for up to 64 CPU cores is plotted by Figure 7. The figure also shows that the scalability of our computational methods (our simulator) is good. Compared with the Example 6, the scalability of this example is weaker. The reason is that the Parallel cluster is shared with hundreds of researchers and applications compete network, which reduces our scalability.

Example 8 *The example tests the scalability of CPR-FPF. The experiments are run on the Parallel cluster. The problem is the standard black oil case with 100 millions of grid blocks and 300 millions of unknowns. The inexact Newton method is applied and the termination tolerance is* 10^{-3}. *The linear solver is BICGSTAB. The ABF decoupling strategy and potential reordering techniques are enabled. The number of maximal inner iterations is 100. The overlap for the RAS method is fixed at 1 and ILUT is used for each subdomain problem. Again, the simulation period is fixed at 20 time steps again. The experiments are performed on the Parallel from University of Calgary, and up to 3,072 MPI processors are employed. Summaries of numerical results are shown in Table 27 and Fig. 8.*

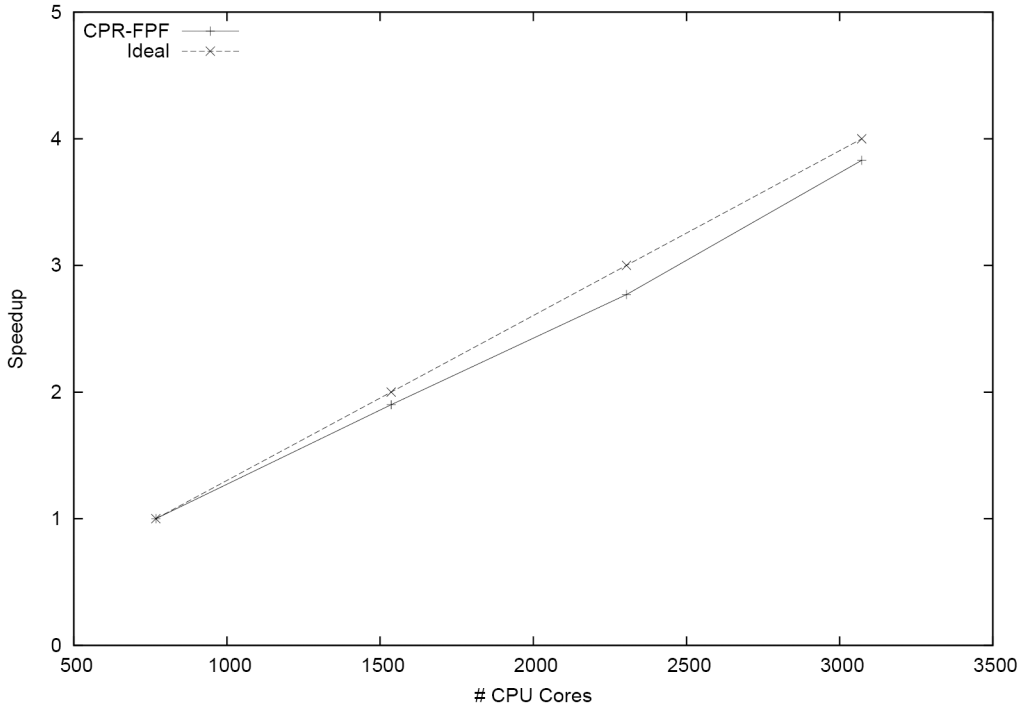

Figure 8—Speedup of CPR-FPF, Example 8

Table 27—Summary of the CPR-FPF preconditioner, Example 8

# Procs	# Newton	# Solver	Time (min)	Speedup
768	42	84	42.1	1.
1536	42	84	22.1	1.90
2304	42	84	15.2	2.77
3072	42	84	11.0	3.83

From Table 27, we can see that the CPR-FPF is very effective and it is an efficient preconditioner for giant reservoir simulations. Even working with a model with 100 millions of grid blocks and 300 millions of unknowns, the linear solver BICGSTAB with the CPR-FPF preconditioner converges in two iterations. The preconditioner is also stable and not sensitive to the number of MPI processors. In the case of 3,072 CPU cores, the linear solver also converges in two iterations. The scalability of our linear solver and preconditioner is demonstrated again. Here the case of 768 CPU cores is the base case and its speedup is 1. When using 1536 CPU cores, the speedup is 1.90 while the ideal speedup is 2. The efficiency is 95%. When using 3,072 CPU cores, the speedup is 3.83 while the ideal is 4, and the efficiency is 95.75%. This example shows our linear solver and preconditioner are effective and scalable, which is also demonstrated in Fig 8. From this figure, we can see the real speedup of this example is very close to the ideal speedup. We can conclude that our preconditioner is efficient for extremely large reservoir simulations.

Conclusions

The solution techniques for the black oil model are studied in this paper, several physics-based preconditioners are developed, and the ILU methods employed in the classical CPR preconditioner are replaced by the RAS method. When one processor is used, the RAS method is equivalent to the ILU methods. We can say that the preconditioners proposed in this paper are suitable for both sequential and parallel programs. In the numerical experiments section, several experiments are calculated, which show that these preconditioners are efficient, stable and scalable. For difficult problems, the average number of iterations

for GMRES is fewer than 60, and the average number of iterations for BICGSTAB is fewer than 10. When calculating simple models, the average number of iterations is around 2. The results show our preconditioners are effective for both the black oil model with highly heterogeneous geological properties and simple models. The results also indicate that BICGSTAB is more stable than GMRES(m) for the black oil model.

Acknowledgements

The support of Department of Chemical and Petroleum Engineering, University of Calgary and Reservoir Simulation Group is gratefully acknowledged. The research is partly supported by NSERC/AIEES/ Foundation CMG and AITF (iCORE) Chairs.

References

1. R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. Van der Vorst, *Templates for the solution of linear systems: building blocks for iterative methods,* 2nd Edition, SIAM, 1994.
2. K.G. Li, A.H. Dogru, A.E. McDonald, A.R. Merchant, A.A. Al-Mulhem, S.B. Al-Ruwaili, N.A. Sobh, and H.A. Al-Sunaidi, Improving the Performance of Mars Simulator on Cray-2 Supercomputer. SPE 29856 presented at the Middle East Oil Show in Bahrain, 11-14 March 1995.
3. J. A. Wheeler, and R. A. Smith, Reservoir Simulation on a Hypercube. SPE 19804 presented at the 64th Annual SPE Conference & Exhibition, SanAntonio, October 1989.
4. J. E. Killough, and R. Bhogeswara, Simulation of Compositional Reservoir Phenomena on a Distributed Memory Parallel Computer, *J. Pet. Tech.,* Nov. 1991.
5. J. E. Killough and Rao Bhogeswara. Simulation of compositional reservoir phenomena on a distributed-memory parallel computer. *Journal of Petroleum Technology* **43.11** (1991): 1368–1374.
6. J. M. Rutledge, D. R. Jones, W. H. Chen, and E. Y Chung, The Use of Massively Parallel SIMD Computer for Reservoir Simulation. SPE Paper 21213 presented at the eleventh SPE Symposium on Reservoir Simulation, Anaheim, 1991.
7. Gautam Shiralkar, R.E. Stephenson, Wayne Joubert, Olaf Lubeck, and Bart van Bloemen Waanders. A production quality distributed memory reservoir simulator. SPE Reservoir Simulation Symposium. 1997.
8. Terje Kaarstad, Johnny Froyen, Petter Bjorstad, Magne Espedal, *Massively Parallel Reservoir Simulator,* SPE-29139, San Antonio, Texas, 1995.
9. John E. Killough, Dominic Camilleri, Bruce L. Darlow, John A. Foster, *Parallel Reservoir Simulator Based on Local Grid Refinement,* SPE-37978, Dallas, 1997.
10. A. H. Dogru, H. A. Sunaidi, L. S. Fung, W. A. Habiballah, N. Al-Zamel, K. G. Li, A parallel reservoir simulator for large-scale reservoir simulation. *SPE Reservoir Evaluation & Engineering* **5.1** (2002): 11–23.
11. Ali H. Dogru, Larry Siu Kuen Fung, Usuf Middya, Tareq Al-Shaalan, Jorge Alberto Pita, A next-generation parallel reservoir simulator for giant reservoirs. SPE/EAGE Reservoir Characterization & Simulation Conference. 2009.
12. Manish Parashar, and Ivan Yotov, An environment for parallel multi-block, multi-resolution reservoir simulations, Proceedings of the 11th International Conference on Parallel and Distributed Computing Systems (PDCS 98), Chicago, IL, International Society for Computers and their Applications (ISCA), 1998.
13. Yousef Saad, *Iterative methods for sparse linear systems*. Siam, 2003.
14. J. R. Wallis, Incomplete Gaussian elimination as a preconditioning for generalized conjugate gradient acceleration. SPE Reservoir Simulation Symposium. 1983.

15. G. Alda Behie, and P. A. Forsyth, Jr., Incomplete factorization methods for fully implicit simulation of enhanced oil recovery. *SIAM Journal on Scientific and Statistical Computing* **5.3** (1984): 543–561.
16. J. R. Wallis, R. P. Kendall, and T. E. Little. Constrained residual acceleration of conjugate residual methods. SPE Reservoir Simulation Symposium. 1985.
17. Hui Cao, Hamdi A. Tchelepi, John Richard Wallis, Hrant E. Yardumian, Parallel scalable unstructured CPR-type linear solver for reservoir simulation. SPE Annual Technical Conference and Exhibition. 2005.
18. R. E. Bank, T. F. Chan, W. M. Coughran, Jr., R. K. Smith, The Alternate-Block-Factorization procedure for systems of partial differential equations. *BIT Numerical Mathematics* **29.4** (1989): 938–954.
19. K. Wu, X. Li, and Y. Zhai, The Model for Predicting Stream Breakthrough Timing during Steam Drive Development of Heavy Oil Reservoirs. Paper SPE 150504-MS presented at SPE Heavy Oil Conference and Exhibition in Kuwait City, Kuwait, 12-14 December, 2011
20. Sebastien Lacroix, Yuri V. Vassilevski, and Mary F. Wheeler. Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS). *Numerical linear algebra with applications* **8.8** (2001): 537–549.
21. K. Wu, X. Li, C. Wang, W. Yu, C. Guo, D. Ji, G. Ren, Z. Chen, Apparent Permeability for Gas Flow in Shale Reservoirs Coupling Effects of Gas Diffusion and Desorption. Paper SPE-1921039-MS presented at SPE/AAPG/SEG Unconventional Resources Technology Conference, 25-27 August, Denver, Colorado, USA, 2014.
22. R. Scheichl, R. Masson, and J. Wendebourg, Decoupling and Block Preconditioning for Sedimentary Basin Simulations, *Computational Geosciences,* Vol. **7,** 2003: 295–318.
23. Tareq Mutlaq Al-Shaalan, Hector Manuel Klie, Ali H. Dogru, Mary Fanett Wheeler, Studies of Robust Two Stage Preconditioners for the Solution of Fully Implicit Multiphase Flow Problems. SPE Reservoir Simulation Symposium. 2009.
24. K. Wu, X. Li, Z. Chen. 2015. A General Model of Gas Flow in Shale Reservoirs with Nano-to Micro-scales. Paper SPE-173201-MS will be presented at SPE Reservoir Simulation Symposium, Houston, TX, and February 23-25, 2015.
25. Teng Chen, Nicholas Gewecke, Zhen Li, Andrea Rubiano, Robert Shuttleworth, Bo Yang and Xinghui Zhong, *Fast Computational Methods for Reservoir Flow Models,* 2009.
26. Andrea Elli, and Olof B. Widlund. *Domain decomposition methods: algorithms and theory*. Vol. **34**. Springer, 2005.
27. K. Wu, X. Li, H., W. Guan, X. Wang, Z. Liao and W. Li. A quantitative model for evaluating the impact of volatile oil non-equilibrium phase transition on degassing [J]. *Petroleum Exploration & Development,* 2012, **39**(5):597–604.
28. Xiao-Chuan Cai, and Marcus Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing* **21.2** (1999): 792–797.
29. K. Wu, X. Li, B. Yan, J. Shi, Q. Li. Research on computational method for reservoir pressure of water-drive condensate gas reservoir. *Petroleum Science and Technology,* **31**:17, 1744–1751, 2013.
30. H. Liu, S. Yu, Z. Chen, B. Hsieh and L. Shao, *Parallel Preconditioners for Reservoir Simulation on GPU,* LACPEC/SPE-152811.
31. H. Liu, S. Yu, Z. Chen, B. Hsieh and L. Shao, Sparse Matrix-Vector Multiplication on NVIDIA GPU, *International Journal of Numerical Analysis & Modeling,* Series B, 2012, Volume **3,** No. 2: 185–191.
32. H. Klie, H. Sudan, R. Li, and Y. Saad, Exploiting capabilities of many core platforms in reservoir simulation, SPE RSS Reservoir Simulation Symposium, 21-23 February 2011.

33. R. Li and Y. Saad, *GPU-accelerated preconditioned iterative linear solvers,* Technical Report umsi-2010-112, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2010.
34. S. Yu, H. Liu, Z. Chen, B. Hsieh, and L. Shao, GPU-based Parallel Reservoir Simulation for Large-scale Simulation Problems, SPE Europec/EAGE Annual Conference, Copenhagen, Denmark, 2012.
35. Z. Chen, H. Liu, S. Yu, B. Hsieh, and L. Shao, Reservoir Simulation on NVIDIA Tesla GPUs, The Eighth International Conference on Scientific Computing and Applications, University of Nevada, Las Vegas, April, 2012.
36. Z. Chen, H. Liu, S. Yu, B. Hsieh and L. Shao, GPU-based parallel reservoir simulators, 21st International Conference on Domain Decomposition Methods, 2012, France.
37. Z. Chen, H. Liu and B. Yang, Parallel triangular solvers on GPU, Proceedings of International Workshop on Data-Intensive Scientific Discovery (DISD), Shanghai University, Shanghai, China, August 1-4, 2013.
38. K. Stüben, T. Clees, H. Klie, B. Lou, M. F. Wheeler, Algebraic multigrid methods (AMG) for the efficient solution of fully implicit formulations in reservoir simulation. SPE Reservoir Simulation Symposium. 2007.
39. H. Liu, Z. Chen and B. Yang, Accelerating preconditioned iterative linear solver on GPU, *International Journal of Numerical Analysis and Modelling: Series B,* **5**(1-2), 2014, 136–146.
40. H. Liu, Z. Chen, S. Yu, B. Hsieh and L. Shao, Development of a Restricted Additive Schwarz preconditioner for Sparse Linear Systems on NVIDIA GPU, *International Journal of Numerical Analysis and Modelling: Series B* **5**(1-2), 2014, 13–20.
41. Z. Chen, H. Liu, and B. Yang, Accelerating iterative linear solvers using multiple graphical processing units, *International Journal of Computer Mathematics,* to appear.
42. H. Deng, Z. Chen, X. Bao, Y. Wei, H. Liu, and C. Dong, Geomechanical and Thermal Simulation of ES-SAGD Process, SPE-148847-MS, Canadian Unconventional Resources Conference, 15-17 November, Alberta, Canada, 2011.
43. X. Bao, H. Deng, H. Zhong, H. Liu, Z. Chen, and C. Dong, Coupled Geomechanical and Thermal Simulation of SAGD Process, SPE-165423-MS, SPE Heavy Oil Conference-Canada, 11-13 June, Calgary, Alberta, Canada, 2013.
44. Klaus Stüben, A review of algebraic multigrid. *Journal of Computational and Applied Mathematics* **128.1** (2001): 281–309.
45. H. Liu, S. Yu and Z. Chen, Development of Algebraic Multigrid Solvers Using GPUs, SPE Reservoir Simulation Symposium, Feb 2013, Houston, USA.
46. Z. Chen, G. Huan, and Y. Ma. *Computational methods for multiphase flows in porous media*. Vol. **2**. Siam, 2006.
47. F. Kwok and H. Tchelepi, Potential-based reduced newton algorithm for nonlinear multiphase flow in porous media. *Journal of Computational Physics* **227.1** (2007): 706–727.
48. *HYPRE: high performance preconditioner*. http://acts.nersc.gov/hypre/
49. Robert D. Falgout, and Ulrike Meier Yang. *hypre: A library of high performance preconditioners*., Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002. 632–641.
50. E. Boman and K. Devine and R. Heaphy and B. Hendrickson and V. Leung and L.A. Riesen and C. Vaughan and U. Catalyurek and D. Bozdag and W. Mitchell and J. Teresco, *Zoltan v3: Parallel Partitioning, Load Balancing and Data-Management Services, User's Guide,* Sandia National Laboratories Tech. Rep. SAND2007-4748W, 2007.
51. H. Liu, Dynamic Load Balancing on Adaptive Unstructured Meshes, 10th IEEE International Conference on High Performance Computing and Communications, 2008.

52. H. Liu and L. Zhang, Existence and construction of Hamiltonian paths and cycles on conforming tetrahedral meshes, *Int. J. Comput. Math.* **88**(6), 2011: 1137–1143.
53. H. Liu, Z. Chen, L. Zhang: Parallel construction of Hamiltonian paths for conforming tetrahedral meshes. *Int. J. Comput. Math.* **90**(7), 2013: 1366–1372.
54. A. Odeh, Comparison of solutions to a three-dimensional black-oil reservoir simulation problem (includes associated paper 9741), *Journal of Petroleum Technology* **33.1** (1981): 13–25.
55. K. Wu, X. Li, P. Yang, S. Zhang. The Establishment on Novel Deliverability Equation of Abnormal Pressure Gas Reservoirs Considering Variable Threshold Pressure Drop. *Petroleum Science and Technology,* **32**:1, 15–21, 2014.
56. K. Wu, X. Li. A New Method to PredictWater Breakthrough Time in EdgeWater Condensate Gas Reservoir Considering Retrograde Condensation. *Petroleum Science and Technology,* **31**:17, 1738–1743, 2013.
57. M. A. Christie, and M. J. Blunt. Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering* **4.4** (2001): 308–317.
58. L. Zhang, A Parallel Algorithm for Adaptive Local Refinement of Tetrahedral Meshes Using Bisection, Numer. *Math.: Theory, Methods and Applications,* 2009, **2,** pp. 65–89.
59. K. Wu, X. Li, X. Wang, B. Yan, Meipeng Ren. Predicting the Method of Oil Recovery in the Gas-assisted Gravity Drainage Process, *Petroleum Science and Technology,* **31**:23, 2527–2533, 2013.
60. L. Zhang, T. Cui, and H. Liu, A set of symmetric quadrature rules on triangles and tetrahedra, *J. Comput. Math,* 2009, **27**(1), pp. 89–96.
61. K. Wu, X. Li, M. Ruan, M. Yu, X. Du, L. Jiang, Q. Li. Dynamic tracking model for the reservoir water flooding of a separated layer water injection based on a well temperature log. *J Petrol Explor Prod Technol,* 2014.
62. Hui Liu and Linbo Zhang, Parallel implementation of commonly used marking strategies in the adaptive finite element toolbox PHG, *Journal on Numerical Methods and Computer Applications,* 2009, **04,** pp. 315–320.
63. K. Wu, X. Li, X. Gou. Research on Variance Principle of Critical Producing Pressure Drop of Horizontal Well in Gas Reservoir with Bottom Water. *Advanced Materials Research*. 674–679, 2013
64. George Karypis, Kirk Schloegel, and Vipin Kumar. *Parmetis: Parallel graph partitioning and sparse matrix ordering library. Version 1.0,* Dept. of Computer Science, University of Minnesota, 1997.
65. Z. Chen, H. Liu, S. Yu, B. Hsieh and L. Shao, GPU-based parallel reservoir simulators, Proc. of 21st International Conference on Domain Decomposition Methods, 2012, France.
66. Z. Chen, *Reservoir Simulation: Mathematical Techniques in Oil Recovery,* Society for Industrial and Applied Mathematics Philadelphia, PA, USA 2007.