Journal of Computational Physics 301 (2015) 19–34

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A scalable parallel black oil simulator on distributed memory parallel computers

Kun Wang, Hui Liu, Zhangxin Chen *

ARTICLE INFO

Article history:
Received 30 June 2014
Received in revised form 25 July 2015
Accepted 11 August 2015
Available online 20 August 2015

Keywords:
Black oil model
Parallel simulator
Preconditioner
Large-scale parallel computing

ABSTRACT

This paper presents our work on developing a parallel black oil simulator for distributed memory computers based on our in-house parallel platform. The parallel simulator is designed to overcome the performance issues of common simulators that are implemented for personal computers and workstations. The finite difference method is applied to discretize the black oil model. In addition, some advanced techniques are employed to strengthen the robustness and parallel scalability of the simulator, including an inexact Newton method, matrix decoupling methods, and algebraic multigrid methods. A new multi-stage preconditioner is proposed to accelerate the solution of linear systems from the Newton methods. Numerical experiments show that our simulator is scalable and efficient, and is capable of simulating extremely large-scale black oil problems with tens of millions of grid blocks using thousands of MPI processes on parallel computers.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

Nowadays, large-scale reservoir simulations often have millions of grid blocks, particularly those involving thermal recoveries such as SAGD (Steam Assisted Gravity Drainage) simulations, which require that grid block sizes are of order of around one meter, which results in an extremely large number of grid blocks for full field scale simulations. Furthermore, when dealing with models with complex geology, high resolution simulations with a large number of grid blocks are required to capture fine-scale phenomenon. Most available commercial reservoir simulators were developed for personal computers and workstations. Limited by the memory size and CPU performance, using these reservoir simulators, simulations with tens of millions of grid blocks may take weeks or even longer to complete. Parallel reservoir simulators should be considered to improve simulation efficiency.

Various techniques have been invested to accelerate reservoir simulations, including new models [1], numerical methods [1], linear solvers and preconditioner techniques [2–7]. In the 1980s, parallel computation was considered in reservoir simulations [8–10]. Killough et al. developed a parallel compositional simulator, which demonstrated that a highly efficient parallel model could be generated for an n-component, three-phase, EOS (equations of state) reservoir simulator in a distributed-memory parallel computer [10]. Rutledge et al. implemented an IMPES (implicit-pressure explicit saturation) compositional simulator using massive SIMD computers, which yielded reasonable computational performance with a straightforward data structure and a simple approach to handle multiple phases [11]. Kaarstad et al. presented a two-dimensional two-phase (oil and water) simulator, which could solve problems with one million grid blocks [12]. Shiralkar et al. developed the parallel production quality simulator FALCON, which could run effectively on a variety of computing platforms [13]. Killough et al. used locally refined grids in their parallel simulator [14]. Parashar et al.'s parallel simulator

* Corresponding author at: Dept. of Chemical and Petroleum Engineering, University of Calgary, Calgary, AB, T2N 1N4, Canada.
E-mail address: zhachen@ucalgary.ca (Z. Chen).

http://dx.doi.org/10.1016/j.jcp.2015.08.016
0021-9991/© 2015 Elsevier Inc. All rights reserved.

successfully handled multiple fault blocks with multiple physics [15]. In 2009, Dogru et al. developed a parallel simulator GigaPOWERS, which was capable of simulating one billion grid blocks [16]. Linear systems from black oil model are ill-conditioned, and proper preconditioners are essential to parallel simulators. Many preconditioners for reservoir simulation have been proposed, including point-wise and block-wise incomplete factorization [3,4,17], domain decomposition [18,17], constrained pressure residual (CPR) [5,6,19], multi-stage [20] and fast auxiliary space preconditioners (FASP) [21,22]. Recently GPU computing techniques have been applied to reservoir simulations. Chen et al. developed GPU-based linear solvers and preconditioners to speed the Krylov subspace linear solvers, algebraic multigrid solvers and preconditioners [23,24]. These techniques have been applied to black oil simulations [25,19]. Klie and Saad implemented their GPU-based linear solvers and applied them to reservoir simulation [26,27].

In this paper, we present our work on developing a parallel black oil simulator. The black oil simulator is based on an in-house parallel platform we are developing, which is designed for general purpose simulators. The black oil model has three mass conservation equations for the water component, the oil component and the gas component. Here the finite difference method is applied to discretize the equations due to its simplicity. The schemes are conservative and an upstream weighting technique is employed for fluid and fluid–rock properties. The black oil model is highly nonlinear. Usually Newton–Raphson methods (or Newton methods) are used to solve the model system, which is solved relatively accurately. The Krylov subspace solvers [28], such as BICGSTAB and GMRES, and algebraic multigrid solvers (AMG) are widely used to solve linear systems from the Newton methods. The problem is that when the sizes of the linear systems derived from these methods are large and the condition numbers of these linear systems are large, the standard Newton method can be very expensive. To our knowledge, when a linear system is large enough, 70% or more of the whole simulation time can be spent on linear solvers [1]. In our simulator, both the Newton methods and inexact Newton methods are implemented. For the inexact Newton methods, the termination tolerance for the linear solvers is loosed, which saves the simulation time. Even with these methods, the solution of a large linear system in reservoir simulation is still challenging. Commonly used preconditioners are implemented to accelerate the linear solvers, including domain decomposition, algebraic multigrid and Constrained Pressure Residual (CPR) preconditioners. Here, a new three-stage preconditioner is proposed, which is more efficient than the classical CPR preconditioner. The number of average linear iterations is around 10 for the challenging problems studied. It also shows excellent performance and parallel scalability. Numerical experiments are carried to study the performance of the black oil simulator, linear solvers, preconditioners and Newton methods. The benchmarks demonstrate that our methods are fast, our simulator is scalable, and our linear solver techniques are efficient and robust.

The rest of the paper is organized as follows: In Section 2, the mathematical equations of the black oil model and the numerical methods used to discretize these equations are presented. In Section 3, the methods of solving the coupled non-linear systems for the black oil model are introduced. In Section 4, techniques for the solution of the linear systems resulted from Newton methods are studied. In Section 5, numerical results are presented.

2. Black oil model and its numerical discretization

The black oil model assumes that the flow in a reservoir has three phases and three components (oil, gas and water), there is no mass transfer between the water phase and the other two phases, and the reservoir is isothermal. Combining Darcy's law and mass conservation law, the black oil model is written as follows [1]:

$$\begin{cases} \dfrac{\partial}{\partial t}(\phi s_o \rho_o^o) & = \nabla \cdot (\dfrac{K K_{ro}}{\mu_o} \rho_o^o \nabla \Phi_o) + q_o, \\ \dfrac{\partial}{\partial t}(\phi s_w \rho_w) & = \nabla \cdot (\dfrac{K K_{rw}}{\mu_w} \rho_w \nabla \Phi_w) + q_w, \\ \dfrac{\partial (\phi \rho_o^g s_o + \phi \rho_g s_g)}{\partial t} & = \nabla \cdot (\dfrac{K K_{ro}}{\mu_o} \rho_o^g \nabla \Phi_o) + \nabla \cdot (\dfrac{K K_{rg}}{\mu_g} \rho_g \nabla \Phi_g) + q_o^g + q_g, \end{cases} \tag{1}$$

where ϕ and K are porosity and permeability, for phase α ($\alpha = o, w, g$), Φ_α is potential, and s_α, μ_α, p_α, B_α, ρ_α, $K_{r\alpha}$ and q_α are the saturation, viscosity, pressure, formation volume factor, density, relative permeability and production rate, respectively. ρ_o^g is the density of solution gas, ρ_o^o is the density of the oil component, and $\rho_o = \rho_o^g + \rho_o^o$. These variables have the following relations:

$$\begin{aligned} &\Phi_\alpha = p_\alpha + \rho_\alpha \wp z, \\ &s_o + s_w + s_g = 1, \\ &p_w = p_o - p_{cow}(s_w), \\ &p_g = p_o + p_{cog}(s_g), \end{aligned}$$

where $\wp$ is the gravitational constant and z is the reservoir depth. With proper boundary and initial conditions, a close system is given.

In this paper, the fully implicit method (FIM) and a first-order upstream finite difference method are applied to discretize the black oil model, where pressure δp, water saturation δs_w and δX (gas saturation δs_g or bubble point pressure δp_b) are

chosen as the primary variables. The well flow rate constraints are modeled by the Peaceman model [1] and they are treated implicitly. For convenience of description, the contribution of well constraints to the black oil model will be ignored for the rest of this paper. The time term is discretized by the backward Euler difference scheme and the space terms are discretized by the cell-centered finite difference method [1]. Let f^n represent the value of variable f at time step n, and then its derivative at any given time step $(n+1)$ can be approximated by

$$(\frac{\partial f}{\partial t})^{n+1} = \frac{f^{n+1} - f^n}{\Delta t}. \tag{2}$$

Let d be a space direction and A be the area of the corresponding face of the grid block. The transmissibility term $T_{\alpha,d}$ can be written as

$$T_{\alpha,d} = \frac{K K_{r\alpha}}{\mu_\alpha} \rho_\alpha \frac{A}{\Delta d}. \tag{3}$$

For any grid block (i, j, k), we define

$$\begin{aligned} \Delta T_\alpha \Delta \Phi_\alpha \equiv\ & (T_{\alpha,x})_{i+\frac{1}{2},j,k}(\Phi_{i+1,j,k} - \Phi_{i,j,k}) - (T_{\alpha,x})_{i-\frac{1}{2},j,k}(\Phi_{i-1,j,k} - \Phi_{i,j,k}) \\ & + (T_{\alpha,y})_{i,j+\frac{1}{2},k}(\Phi_{i,j+1,k} - \Phi_{i,j,k}) - (T_{\alpha,y})_{i,j-\frac{1}{2},k}(\Phi_{i,j-1,k} - \Phi_{i,j,k}) \\ & + (T_{\alpha,z})_{i,j,k+\frac{1}{2}}(\Phi_{i,j,k+1} - \Phi_{i,j,k}) - (T_{\alpha,z})_{i,j,k-\frac{1}{2}}(\Phi_{i,j,k-1} - \Phi_{i,j,k}). \end{aligned} \tag{4}$$

Then the nonlinear system at the grid block (i, j, k) is

$$\begin{cases} \frac{V}{\Delta t}[(\phi\rho_o^o s_o)^{n+1} - (\phi\rho_o^o s_o)^n] = \Delta T_{o,o}^{n+1} \Delta\Phi_o^{n+1} + q_{o,o}^{n+1} \\ \frac{V}{\Delta t}[(\phi\rho_w s_w)^{n+1} - (\phi\rho_w s_w)^n] = \Delta T_w^{n+1} \Delta\Phi_w^{n+1} + q_w^{n+1} \\ \frac{V}{\Delta t}[(\phi\rho_o^g s_o + \phi\rho_g s_g)^{n+1} - (\phi\rho_o^g s_o + \phi\rho_g s_g)^n] \\ \qquad = \Delta T_{o,g}^{n+1} \Delta\Phi_o^{n+1} + \Delta T_g^{n+1} \Delta\Phi_g^{n+1} + q_{o,g}^{n+1} + q_g^{n+1}, \end{cases} \tag{5}$$

where V is the volume of the grid block at (i, j, k). An upstream-type method is employed in our simulator, which means $(T_{\alpha,x})_{i\pm\frac{1}{2},j,k}$ in formula (4) is calculated as

$$(T_{\alpha,x})_{i\pm\frac{1}{2},j,k} = \begin{cases} (T_{\alpha,x})_{i\pm1,j,k}, & \text{if } \Phi_{i\pm1,j,k} \geq \Phi_{i,j,k} \\ (T_{\alpha,x})_{i,j,k}, & \text{if } \Phi_{i\pm1,j,k} < \Phi_{i,j,k} \end{cases}. \tag{6}$$

Likewise, $(T_{\alpha,y})_{i,j\pm\frac{1}{2},k}$ and $(T_{\alpha,z})_{i,j,k\pm\frac{1}{2}}$ can be calculated by the similar rules.

The nonlinear system can be represented by

$$F(x^*) = 0, \tag{7}$$

where

$$x^* = \begin{pmatrix} s_w \\ X \\ p \end{pmatrix}, \tag{8}$$

and $X = s_g$ or p_b. Methods for solving this nonlinear system will be studied in the following sections.

3. Nonlinear methods for the black oil model

In this section, methods for solving nonlinear systems are presented, including the Newton method and an inexact Newton method. Related techniques are also studied.

Let x^l be the variable x at the lth iteration of a nonlinear solver, and we have

$$x^{n+1} \approx x^{l+1} = x^l + \delta x. \tag{9}$$

By applying the above equation, the nonlinear system (7) can be written as

$$\begin{cases} \dfrac{V}{\Delta t}[(\phi\rho_o^o s_o)^l - (\phi\rho_o^o s_o)^n + \delta(\phi\rho_o^o s_o)] \\ \qquad = \Delta(T_{o,o}^l + \delta T_{o,o})\Delta(\Phi_o^l + \delta\Phi_o) + q_{o,o}^l + \delta q_{o,o} \\ \dfrac{V}{\Delta t}[(\phi\rho_w s_w)^l - (\phi\rho_w s_w)^n + \delta(\phi\rho_w s_w)] \\ \qquad = \Delta(T_w^l + \delta T_w)\Delta(\Phi_w^l + \delta\Phi_w) + q_w^l + \delta q_w \\ \dfrac{V}{\Delta t}[(\phi\rho_o^g s_o + \phi\rho_g s_g)^l - (\phi\rho_o^g s_o + \phi\rho_g s_g)^n + \delta(\phi\rho_o^g s_o + \phi\rho_g s_g)] \\ \qquad = \Delta(T_{o,g}^l + \delta T_{o,g})\Delta(\Phi_o^l + \delta\Phi_o) + \Delta(T_g^l + \delta T_g)\Delta(\Phi_g^l + \delta\Phi_g) + q_{o,g}^l + q_g^l. \end{cases} \tag{10}$$

A linear system $J\delta x = b(x^l)$ is obtained, where J is the Jacobian matrix derived from (10), $b(x^l)$ is the residual and δx is the unknown to be solved for. Applying a proper matrix reordering technique, the Jacobian matrix J has a structure of

$$J = \begin{pmatrix} J_{ws_w} & 0 & J_{wp} \\ J_{os_w} & J_{oX} & J_{op} \\ J_{gs_w} & J_{gX} & J_{gp} \end{pmatrix}, \tag{11}$$

where matrix $J_{\alpha s_w}$ $(\alpha = w, o, g)$ is the submatrix corresponding to the water saturation unknowns, matrix $J_{\alpha X}$ $(\alpha = o, g)$ is the submatrix corresponding to the gas saturation or bubble point pressure unknowns and matrix $J_{\alpha p}$ $(\alpha = w, o, g)$ is the matrix corresponding to the pressure unknowns. The solution of the Newton method process is described as in Algorithm 1. An initial guess is set at the beginning of the Newton method, and then a linear system is derived from the initial guess. The linear system is calculated by linear solvers, such as GMRES. After this, the initial guess is updated. Repeat this process until the stop criterion is achieved. The stop criterion for linear solvers is strict for standard Newton methods.

Algorithm 1 The Newton method.

1: Given an initial guess x^0 and stopping criterion ϵ, let $l = 0$ and assemble the right-hand side $b(x^l)$.
2: **while** $\|b(x^l)\| \geq \epsilon$ **do**
3: Assemble the Jacobian matrix J and vector b.
4: Solve the following linear system and get the solution δx:

$$J\delta x = b(x^l) \tag{12}$$

5: Let $l = l + 1$, $x^l = x^{l-1} + \delta x$ and assemble the right-hand side $b(x^l)$.
6: **end while**
7: $x^* = x^l$ is the solution of the nonlinear system.

3.1. An inexact Newton method

As we discussed above, the Newton method solves the linear system accurately, which is expensive and may slow down simulators. Based on our experience, when a time step is large and the initial guess is far from the real solution, there is no need to solve the linear systems accurately. An inexact Newton method is applied, which solves the linear systems approximately. Its algorithm is described in Algorithm 2.

Algorithm 2 The inexact Newton method.

1: Given an initial guess x^0 and stopping criterion ϵ, let $l = 0$ and assemble the right-hand side $b(x^l)$.
2: **while** $\|b(x^l)\| \geq \epsilon$ **do**
3: Assemble the Jacobian matrix J.
4: Find θ_l and x such that

$$\left\|b(x^l) - J\delta x\right\| \leq \theta_l \left\|b(x^l)\right\|. \tag{13}$$

5: Let $l = l + 1$ and $x^l = x^{l-1} + \delta x$.
6: **end while**
7: $x^* = x^l$ is the solution of the nonlinear system.

The algorithm is similar to the standard Newton methods. The only difference is how to choose θ_l. The Newton method chooses zero or a small value; in this case, the solution of the corresponding linear system is accurate. The inexact Newton method has several different choices for θ_l, three of which are listed as follows:

$$\theta_l = \begin{cases} \dfrac{\|b(x^l) - r^{l-1}\|}{\|b(x^{l-1})\|}, \\ \dfrac{\|b(x^l)\| - \|r^{l-1}\|}{\|b(x^{l-1})\|}, \\ \gamma \left(\dfrac{\|b(x^l)\|}{\|b(x^{l-1})\|} \right)^{\beta}, \end{cases} \tag{14}$$

where r^l is the residual of the lth iteration,

$$r^l = b(x^l) - J\delta x. \tag{15}$$

The third one is employed by our simulator, and based on [29], we set $\gamma = 0.5$ and $\beta = \frac{\sqrt{5}+1}{2}$.

3.2. Initial guess

The Newton method is quadratic in convergence if the initial guess is close enough to the real solution. When we solve a time-dependent problem, taking the value of the last time step x^n as the initial guess for the current time step is a proper choice. However, according to our experience, using the extrapolation of the values from the last several time steps achieves better convergence.

In our simulator, we use linear extrapolation. Taking the pressure p as an example, let p^{n-1} and p^n be the pressure at time t_{n-1} and t_n, respectively. The initial guess of pressure at time t_{n+1} is

$$p^{n+1} = p^n + (p^n - p^{n-1}) \frac{t_{n+1} - t_n}{t_n - t_{n-1}}. \tag{16}$$

When the operation conditions of the wells change, the estimated values may fail and give bad initial guess, which results in more nonlinear iterations or even divergence. In our implementation, we employ the extrapolation as the initial guess only when the operation conditions of the wells keep the same as the last time step.

4. Efficient linear solvers for the black oil model

Now, we consider the linear system $Jx = b$. In practice, a preconditioner is always used and the linear system has an equivalent form, $M^{-1}Jx = M^{-1}b$. In the following section, advanced techniques for accelerating the linear solvers will be studied.

4.1. Decoupling operator

In order to weaken the coupling between the pressure unknowns and the saturation or bubble point pressure unknowns, a decoupling operation is applied to the Jacobian matrix J. This decoupling procedure is required to achieve the effectiveness of the multi-stage preconditioner we will develop in the next section. It is also computationally unexpensive. The decoupling procedure process is indispensable to the multi-stage preconditioner.

There are many well-known options, such as a Quasi-IMPES method [30], a BDS method based on least squares, the Householder transformation [31], and the alternative block factorization (ABF) method [7]. A detailed comparison between these methods can be found in [20]. Here we choose the ABF method in our computation, which is written as

$$M_L = \begin{pmatrix} \mathbf{Diag}(J_{ws_w}) & 0 & \mathbf{Diag}(J_{wp}) \\ \mathbf{Diag}(J_{os_w}) & \mathbf{Diag}(J_{oX}) & \mathbf{Diag}(J_{op}) \\ \mathbf{Diag}(J_{gs_w}) & \mathbf{Diag}(J_{gX}) & \mathbf{Diag}(J_{gp}) \end{pmatrix},$$

where $\mathbf{Diag}(A)$ stands for a diagonal matrix. Then we have $\tilde{J} = M_L^{-1}J$. We denote $\tilde{J}$ by

$$\tilde{J} = \begin{pmatrix} \tilde{J_{ws_w}} & \tilde{J_{wX}} & \tilde{J_{wp}} \\ \tilde{J_{os_w}} & \tilde{J_{oX}} & \tilde{J_{op}} \\ \tilde{J_{gs_w}} & \tilde{J_{gX}} & \tilde{J_{gp}} \end{pmatrix},$$

and $\tilde{b} = M_L^{-1}b$. The system $\tilde{J}x = \tilde{b}$ is equivalent to $Jx = b$. In the following sections, we still use the notation $Jx = b$.

4.2. A multi-stage preconditioner

It is well-known that the pressure equation from the black oil model equations is elliptic, while the equations are hyperbolic corresponding to the saturation or bubble point pressure, if the capillary pressures are ignored. In addition, the pressure unknowns dominate the error of the solution to the linear system $Jx = b$. A multi-stage preconditioner is developed to deal with the different blocks stage-by-stage. In each stage, a different solver technique is applied.

We first describe the general formula of the multi-stage preconditioner M^{-1}. Let r^0 be the initial residual:

$$r^0 = b - Jx^0, \tag{17}$$

where x^0 is an initial guess. The residual r^n at the end of the n-stage precondition process can be represented as

$$r^n = (I - M_n^{-1} J) r^{n-1}. \tag{18}$$

Then the general formula is

$$\begin{aligned} M^{-1} = {} & M_n^{-1}(I - M_{n-1}^{-1} J) \cdots (I - M_1^{-1} J) \\ & + \cdots \\ & + M_2^{-1}(I - M_1^{-1} J) \\ & + M_1^{-1}, \end{aligned} \tag{19}$$

where M_i^{-1} is the ith stage preconditioner. The constrained pressure residual (CPR) method [5,6] and fast auxiliary space preconditioner (FASP) [21,22] are examples of this type of preconditioners. The CPR method separates the pressure block from the full system and uses the AMG method to solve the pressure block, which is the first stage of CPR. The second stage of CPR uses a global smoother (such as the ILU method) to the full system. The FASP method adds two stages to deal with the saturation block resulted from the saturation unknowns and the well block resulted from the well unknowns.

In [21,22], the FASP preconditioner has been compared with CPR preconditioner, and the numerical experiments show the FASP preconditioner is more effective than the CPR preconditioner. However, if the inexact Newton method is employed, the linear iterations are not necessary to converge to a very small tolerance. The computational cost of the extra two stages in the FASP method may cause the FASP preconditioned linear solver spend more time than the CPR preconditioned linear solver, even if the FASP preconditioned linear solver has fewer iterations. In this paper, a three-stage preconditioner is developed to conquer the disadvantages of the FASP preconditioner. In the FASP preconditioner, one of the extra two stages is to gather the well block from all the MPI processes into one and solve it by the direct methods. This stage reduces the parallel scalability significantly, so it is removed in our preconditioner. The other additional stage in the FASP preconditioner is to solve the saturation block with one sweep of block Gauss–Seidel iteration with downwind ordering and crosswind blocks or ILU methods. Considering the problem size of the saturation block, the computational cost of assembling the local matrices and the incomplete LU factorization of the local matrices is very expensive. In our three-stage preconditioner, this stage is replaced by a global smoother, which can save the setup time for the incomplete LU factorization and even make the preconditioner more effective.

Here transfer operators Π_p and Π_S are defined as

$$\Pi_p \begin{pmatrix} r_w \\ r_o \\ r_g \end{pmatrix} = r_g, \tag{20}$$

and

$$\Pi_S \begin{pmatrix} r_w \\ r_o \\ r_g \end{pmatrix} = \begin{pmatrix} r_w \\ r_o \end{pmatrix}. \tag{21}$$

The three-stage preconditioner is presented as in Algorithm 3. The CPR preconditioner can be presented as in Algorithm 4. Due to the parallel inefficiency of the well stage in the FASP preconditioner, the FASP preconditioner without the well stage is considered in this paper and can be presented as in Algorithm 5.

Algorithm 3 Three-stage preconditioner.

1: Given an initial guess x^0, find
2: $x^1 = x^0 + R^{-1}(b - Jx^0)$,
3: $x^2 = x^1 + \Pi_p M_p^{-1} \Pi_p^T (b - Jx^1)$,
4: $x^3 = x^2 + R^{-1}(b - Jx^2)$.

Algorithm 4 The CPR preconditioner.

1: Given an initial guess x^0, find
2: $x^2 = x^1 + \Pi_p M_p^{-1} \Pi_p^T (b - Jx^1)$,
3: $x^3 = x^2 + R^{-1}(b - Jx^2)$.

Algorithm 5 The FASP preconditioner.

1: Given an initial guess x^0, find
2: $x^1 = x^0 + \Pi_S S^{-1} \Pi_S^T (b - Jx^0)$,
3: $x^2 = x^1 + \Pi_p M_p^{-1} \Pi_p^T (b - Jx^1)$,
4: $x^3 = x^2 + R^{-1}(b - Jx^2)$.

The preconditioner R^{-1} solves the full system. The LSOR method [21], ILU method (or Block ILU method) and block Gauss–Seidel method have been successfully applied. However, these methods are designed as serial programs. We choose the Restricted Additive Schwarz (RAS) method [18] as the preconditioner R^{-1} for the full system. This method is easy to parallelize and has fewer iterations than other domain decomposition methods [18,17]. The RAS method has been used as a default preconditioner in many packages. In our simulator, we employ the ILU(0) method as the local solver of RAS. For simplicity, we use RAS-ILU to represent the RAS method with ILU(0) as the local solver in this paper. Details of the RAS method can be found in [32,18,17].

The preconditioner S^{-1} is designed to solve the saturation block, which is

$$\begin{pmatrix} \tilde{J_{ws_w}} & \tilde{J_{wX}} \\ \tilde{J_{os_w}} & \tilde{J_{oX}} \end{pmatrix}.$$

Similar as the global smoother R^{-1}, the RAS-ILU method is employed to solve the saturation block in this paper.

The preconditioner M_p^{-1} is designed to solve the pressure block J_{gp}. This block is elliptic, and the algebraic multigrid (AMG) method is applied. It is not necessary to solve M_p accurately in practice. Only one AMG V-cycle is used and the BoomerAMG from HYPRE [33] is employed as the AMG solver in our computation.

5. Numerical experiments

In this section, we present numerical experiments for our black oil simulator, which is implemented based on PRSI, a parallel general purpose platform for reservoir simulators developed by our group. The PRSI is written by C and MPI (Message Passing Interface), and it provides grids, data, linear solvers, preconditioners, distributed matrices and vectors, visualization, key words parsing and well modeling modules. The grids we support are structured Cartesian grids, which support the finite difference and finite volume methods. The load balancing methods for grid partitioning are the ParMETIS [34], Zoltan [35], and Hilbert space-filling curve methods [36]. The Krylov subspace solvers and AMG solvers are supported, including GMRES(m), BICGSTAB [28] and the classic AMG solver [33]. General purpose preconditioners, including ILU, domain decomposition [18] and AMG [33], and special preconditioners, including CPR [5,6,17,19] and the preconditioner introduced in this paper, are supported. For the well modeling module, a simple Peaceman model [1] is implemented.

Numerical experiments in Sections 5.2, 5.3 and 5.4 are tested on the cluster GPC (General Purpose Cluster) from Canada's largest supercomputer centre SciNet [37]. The GPC consists of 3780 nodes (IBM iDataPlex DX360M2) with a total of 30,240 cores (Intel Xeon E5540) at 2.53 GHz, with 16 GB RAM per node (some larger-memory nodes up to 32 GB). Our jobs were run on the nodes with 16 GB memory connected with non-blocking DDR InfiniBand. The 16 GB memory and 8 cores of each employed node are fully used in our computation, and each MPI process runs on a core. The case in Section 5.5 is tested on Blue Gene/Q from IBM. It has two racks, each of which has two mid-planes. Each mid-plane has 16 computer nodes, and each node contains 32 compute cards (64-bit PowerPC A2 processor). One compute card has one core for the operation system and 16 cores for computation. The IBM Blue Gene/Q system has 32,768 CPU cores for computation. The 64-bit PowerPC A2 processor cores are 4-way simultaneously multi-threaded, and run at 1.6 GHz. Each MPI process also runs on a core.

Before presenting the numerical experiments, we first introduce the stopping criteria of Newton iterations. At convergence of Newton iterations, the following condition should be satisfied:

- The scaled mass balance errors on each grid block are less than 10^{-3}. The scaled mass balance errors e_α for the phase α ($\alpha = o, w, g$) are defined as

$$e_o = \frac{\frac{V}{\Delta t}[(\phi\rho_o^o s_o)^{n+1} - (\phi\rho_o^o s_o)^n] - \Delta T_{o,o}^{n+1} \Delta\Phi_o^{n+1} - q_{o,o}^{n+1}}{\frac{V}{\Delta t}[(\phi\rho_o^o s_o)^{n+1} - (\phi\rho_o^o s_o)^n]} \tag{22}$$

$$e_w = \frac{\frac{V}{\Delta t}[(\phi\rho_w s_w)^{n+1} - (\phi\rho_w s_w)^n] - \Delta T_w^{n+1} \Delta\Phi_w^{n+1} - q_w^{n+1}}{\frac{V}{\Delta t}[(\phi\rho_w s_w)^{n+1} - (\phi\rho_w s_w)^n]} \tag{23}$$

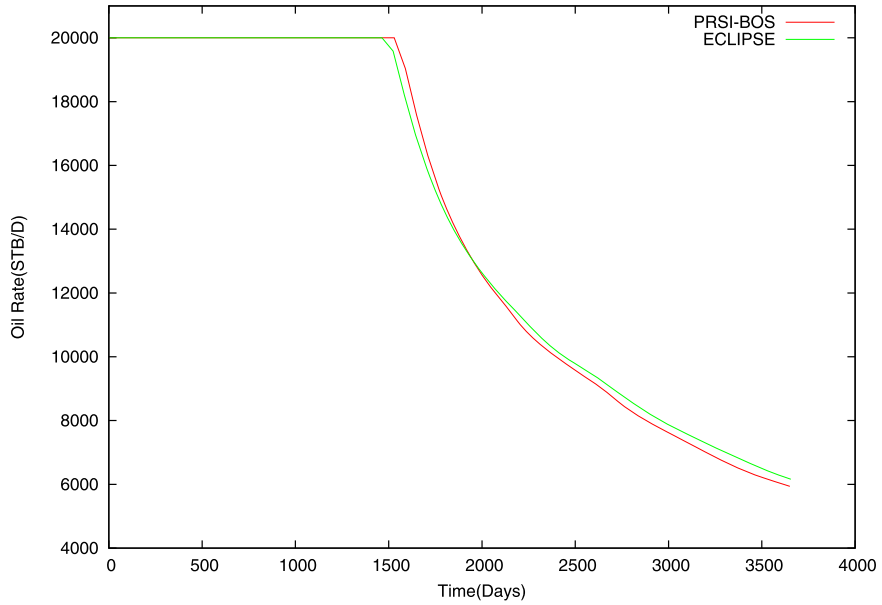

Fig. 5.1. SPE1 case: oil rate vs. time.

$$e_g = \frac{\frac{V}{\Delta t}[(\phi\rho_o^g s_o)^{n+1} - (\phi\rho_o^g s_o)^n] - \Delta T_{o,g}^{n+1}\Delta\Phi_o^{n+1} - q_{o,g}^{n+1}}{\frac{V}{\Delta t}[(\phi\rho_o^g s_o + \phi\rho_g s_g)^{n+1} - (\phi\rho_o^g s_o + \phi\rho_g s_g)^n]} \tag{24}$$

$$+ \frac{\frac{V}{\Delta t}[(\phi\rho_g s_g)^{n+1} - (\phi\rho_g s_g)^n] - \Delta T_g^{n+1}\Delta\Phi_g^{n+1} - q_g^{n+1}}{\frac{V}{\Delta t}[(\phi\rho_o^g s_o + \phi\rho_g s_g)^{n+1} - (\phi\rho_o^g s_o + \phi\rho_g s_g)^n]} \tag{25}$$

- The maximum saturation change is less than 10^{-4}.
- The maximum scaled pressure increment of Newton iteration e_p on each grid block is less than 10^{-4}. The scaled pressure increment of Newton iteration e_p is defined as

$$e_p = \frac{\delta p}{p}. \tag{26}$$

In Tables 1–7, "NP" is the number of MPI processes, "Steps" is the total number of time steps, "Newton" is the total number of Newton iterations, "Linear" is the total number of linear iterations, "A.L.N." is the number of average linear iterations per Newton iteration, "Scalability" is the strong parallel scalability, and "Time" is the total simulation time. The GMRES(50) method is employed as the linear solver.

5.1. SPE1 project

First, we use the SPE (Society of Petroleum Engineers) benchmark problem of Odeh for gas injection to verify our simulator. This is a standard three-phase black oil model, with $10 \times 10 \times 3$ grid blocks and isotropic and layered permeability. One high rate gas injector and one producer are located at two opposite corners. The total simulation time is 10 years. Details of this case can be found in [38].

The simulation results are compared between our simulator and Eclipse by Schlumberger. The oil production rate and gas/oil production rate (GOR) are shown in Figs. 5.1 and 5.2, from which we can see that the results match well and the results of our simulator are correct.

We use this problem as an example to demonstrate the efficiency of the inexact Newton method. As seen in Table 1, the total number of linear iterations using this inexact Newton method is nearly reduced by half compared with the standard Newton method. The inexact Newton method is better than the standard Newton method.

We also take this problem as an example to test the effect of different choices of the initial guess in the inexact Newton method. From Table 2, we can see that the extrapolation gives a much better initial guess.

5.2. SPE10-based black oil case

The Tenth SPE Comparative Project is a challenging test case for linear solvers due to the highly heterogeneous permeability and porosity. The permeability varies between 6.65×10^{-11} and 20 Darcy. Its porosity varies between 0 and 0.5. The

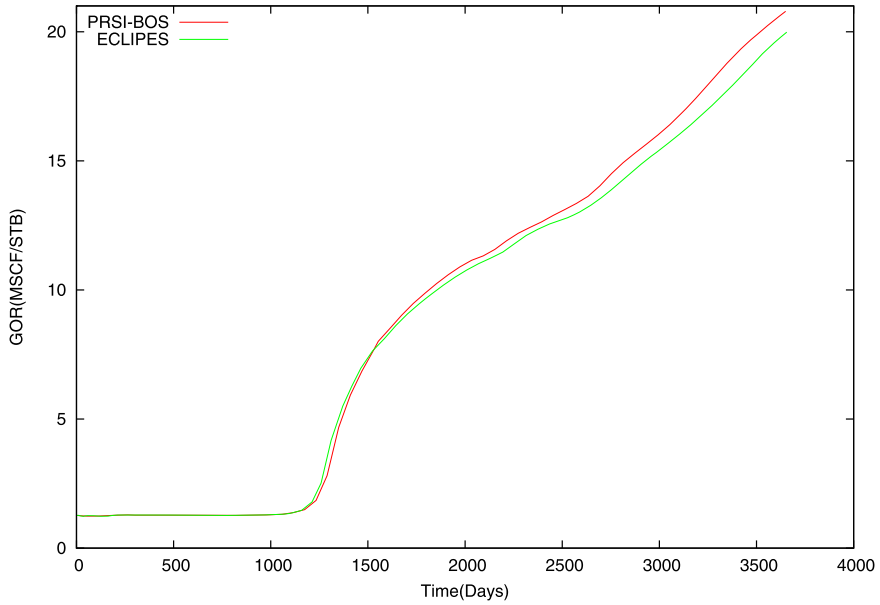

Fig. 5.2. SPE1 case: GOR vs. time.

Table 1
Newton method.

Method	Steps	Newton	Linear
Standard Newton	82	214	549
Inexact Newton	82	223	307

Table 2
Initial guess.

Initial guess	Steps	Newton	Linear
Last time step	82	302	442
Extrapolation	82	223	307

Table 3
Oil and gas PVT table.

Pressure psia	R_s MSCF/STB	B_o RB/STB	μ_o cp	B_g SCF/STB	μ_g cp
400	0.0165	1.012	3.5057	1.96	0.0140
4000	0.1130	1.01278	2.9972	0.84	0.0160
8000	0.1583	1.10759	2.7775	0.59	0.0175
10,000	0.1810	1.155	2.6675	0.42	0.0195

dimensions are $1200 \times 2200 \times 170$ (ft) and have 1.1 million ($60 \times 220 \times 85$) grid blocks (1.09 million of them are active). The top 35 layers (70 ft) represent the Tarber formation and the bottom 50 layers (100 ft) represent the Upper Ness formation. Four producers are placed at four corners of the reservoir, and one injector in the center. The total simulation time is 2000 days. The details can be found in [39].

Fluid properties are adopted to the original SPE10 problem; see Table 3. R_s is the solution gas–oil ratio for saturated oil. The compressibility of undersaturated oil C_o is constant 1.1388×10^{-6} psia^{-1}, and the oil viscosity compressibility of undersaturated oil V_{co} is set to zero. The water formation volume factor is calculated by

$$B_w = \frac{B_w^0}{1 + C_w(p - p^0)}, \tag{27}$$

where the reference formation volume factor $B_w^0 = 1.01$, water compressibility $C_w = 3 \times 10^{-6}$ psia^{-1} and the reference pressure $p^0 = 6000$ psia. The water viscosity $\mu_w = 0.3$ cp is constant.

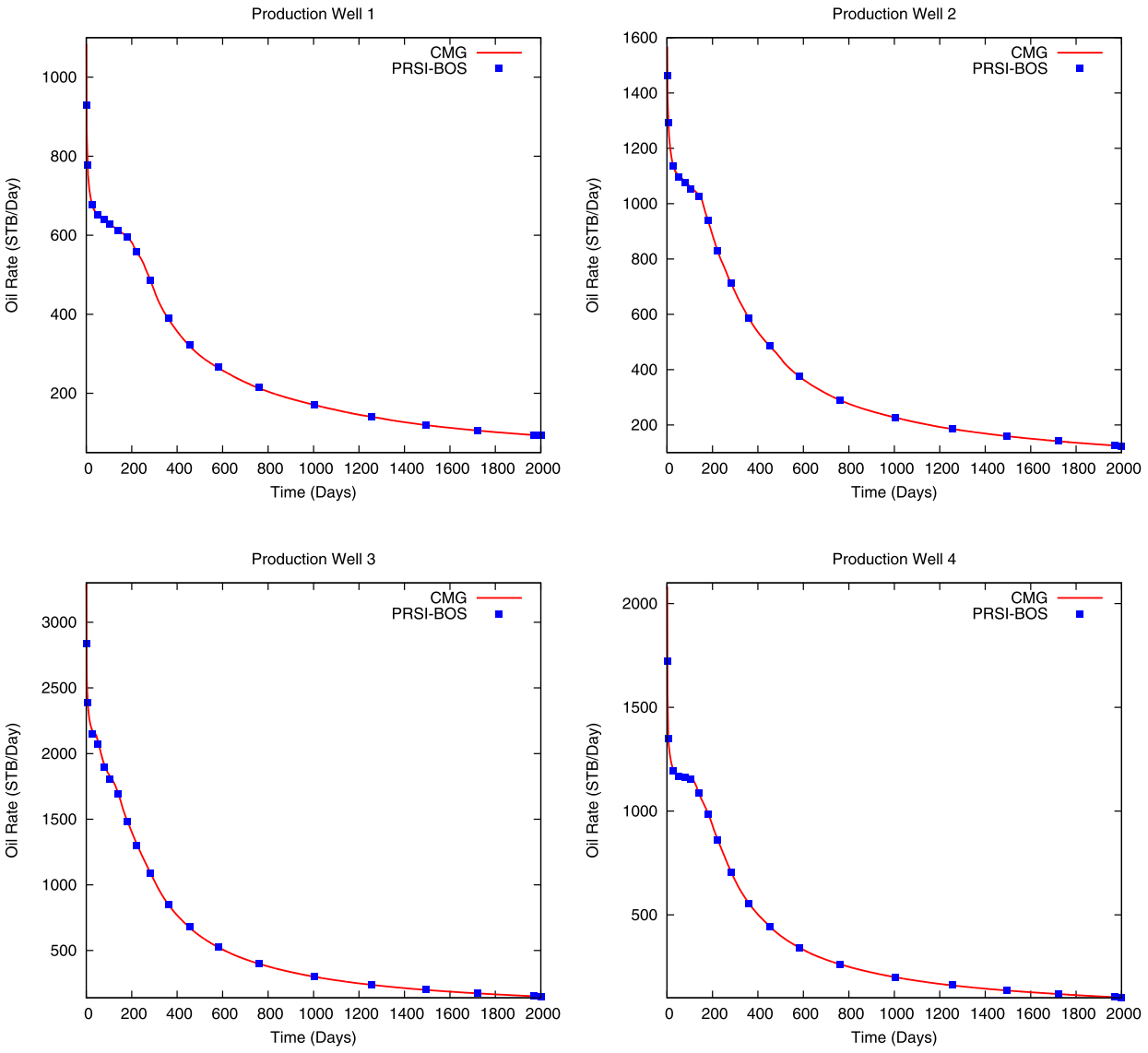

Fig. 5.3. Production rate.

The Jacobian matrices have 3.3 millions unknowns and we use this example to illustrate the robustness of our linear solvers and the parallel scalability of our simulator. The simulation results are compared between our simulator and IMEX by CMG. The oil production rates of four producers are shown in Fig. 5.3. The results are matched well.

The performance of different preconditioners, including the three-stage preconditioner, the CPR preconditioner, the FASP preconditioner and the RAS-ILU preconditioner, is summarized in Table 4. For the RAS-ILU preconditioner, the decoupling process is also employ to increase its efficiency. Even though, as the increasing number of the MPI processes used, the grid blocks on each subdomain will be fewer, which causes a dramatical loss of the effectiveness of the RAS-ILU method and leads to an unstable performance. When the number of the MPI processes is increased from 32 to 64, the number of Newton iterations jumps from 1800 to 3083 and the number of linear iterations increase 1.5 times, from which we can conclude that the RAS-ILU preconditioner is not suitable for large scale reservoir simulations. The CPR preconditioner is much more effective and efficient than the RAS-ILU preconditioner, and it has robust performance. The average linear iterations times of the CPR preconditioned GMRES is around 50. The scalability is good when the number of MPI processes in increased form 16 to 128. When more than 128 MPI processes are used, considering only 1.1 million of grid blocks in this case, the effectiveness of the domain decomposition method becomes weak and the communication between MPI processes becomes dominant, which leads to the loss of the parallel scalability. The FASP preconditioner also has robust performance and results in fewer Newton iterations and linear iterations. However, the extra computation of the FASP preconditioner, including the incomplete LU factorization of the saturation block and the solution of the L and U matrices, leads to more

Table 4
Comparison of different preconditioners for SPE10-based black oil problem.

Preconditioner	NP	Time steps	Newton	Linear	A.L.N.	Time (sec)	Scalability
Three stage	16	407	1558	59,859	38.4	33,447	–
	32	407	1556	58,483	37.6	18,686	0.90
	64	407	1555	63,788	41.0	11,953	0.70
	128	407	1563	59,351	38.0	6652	0.63
	256	407	1583	71,794	45.4	4834	0.43
CPR	16	407	1711	81,257	47.5	35,849	–
	32	407	1715	82,378	48.0	21,746	0.82
	64	407	1716	84,941	49.5	13,812	0.65
	128	407	1726	83,533	48.4	8316	0.54
	256	407	1697	84,523	49.8	5538	0.41
FASP	16	407	1580	67,700	42.8	39,728	–
	32	407	1599	69,746	43.6	24,324	0.82
	64	407	1604	74,901	46.7	14,937	0.66
	128	407	1600	71,997	45.0	9626	0.52
	256	407	1654	84,050	50.8	7581	0.33
RAS-ILU	16	408	1851	296,598	160.2	48,960	–
	32	407	1800	278,999	155.0	21,405	1.14
	64	448	3083	392,617	127.3	18,488	0.66
	128	464	3668	490,575	133.7	14,702	0.42
	256	420	2295	262,837	114.5	6952	0.44

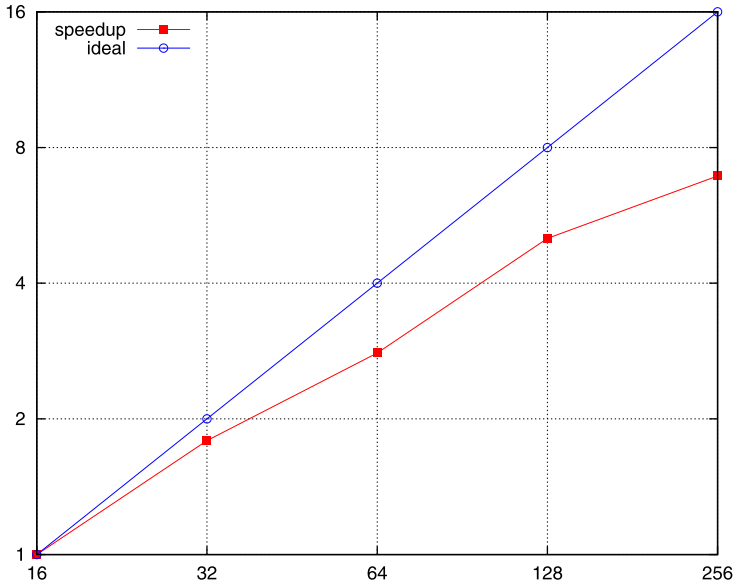

Fig. 5.4. Speedup of SPE10-based case.

computational time. Compared with the CPR and FASP preconditioners, the three-stage preconditioner uses a global RAS-ILU stage to replace the stage for saturation block, which can save the expensive setup time and make the preconditioner more effective. About 10% of the Newton iterations and 25% of the linear iterations are reduced from the CPR preconditioned method because of the extra RAS-ILU stage in the three-stage preconditioner. Due to the big amount of reduction in linear iterations, less total simulation time is needed by the three-stage preconditioner, which demonstrates the efficiency of our preconditioner. The parallel scalability is also improved by the three-stage preconditioner. The speedup of using the three-stage preconditioner is shown in Fig. 5.4.

5.3. *Refined SPE10-based black oil case*

In this case, each grid block of SPE10-based black oil case shown in Section 5.2 is refined into 27 grid blocks, which results in $180 \times 660 \times 255 = 30,294,000$ grid blocks. Because of the limitation of the computational resource, 300 time steps are simulated in this case. Due to the unstable performance of the RAS-ILU preconditioner, the RAS-ILU preconditioner is not considered in this case. Considering the size of this case, we use from 256 to 1024 MPI processes.

Table 5
Refined SPE10-based black oil case.

Preconditioner	NP	Newton	Linear	A.L.N.	Time (sec)	Scalability
Three stage	256	1230	42,519	34.6	67,322	–
	384	1221	42,744	35.0	43,621	1.03
	512	1222	42,188	34.5	35,692	0.94
	768	1223	43,121	35.3	27,088	0.83
	1024	1213	43,304	35.7	23,612	0.71
CPR	256	1253	49,748	39.7	69,164	–
	384	1251	49,912	39.9	44,285	1.04
	512	1247	49,779	39.9	35,974	0.96
	768	1250	51,178	40.9	29,098	0.79
	1024	1251	51,677	41.3	24,866	0.69
FASP	256	1229	45,245	36.8	85,604	–
	384	1230	45,911	37.3	55,950	1.02
	512	1231	45,060	36.6	42,513	1.01
	768	1231	46,973	38.2	33,246	0.86
	1024	1232	46,891	38.1	28,777	0.74

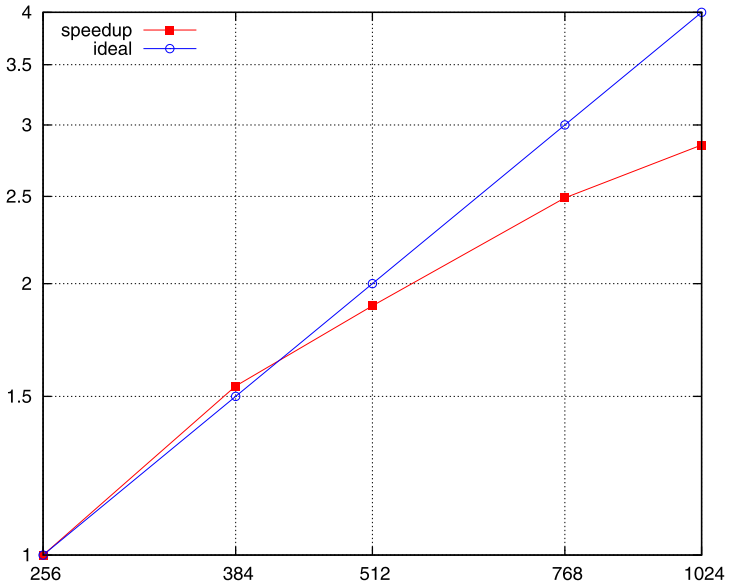

Fig. 5.5. Speedup of refined SPE10-based case.

In Table 5, the three-stage preconditioner, the FASP preconditioner and the CPR preconditioner are compared by detail. The phenomenon is similar as what we observe in the SPE10-based black oil case: The FASP preconditioner linear solver needs fewer linear iterations than the CPR preconditioned one, but the extra computational cost of the FASP preconditioner makes it the slowest method among the three considered methods; From the angle of elapsed simulation time, the CPR preconditioner shows as good performance as the three-stage preconditioner when the number of the MPI processes is less than or equal to 512; As the increasing of the MPI processes, the number of iterations of CPR preconditioned linear solver increases, which results in more elapsed time. The three-stage preconditioned linear solver uses the fewest number of Newton iterations and linear iterations, and has the most stable number of the average linear iterations per Newton iteration; The three-stage preconditioner also needs the least simulation time and gives the best parallel scalability; The speedup of employing the three-stage preconditioned linear solver is shown in Fig. 5.5.

5.4. MX521X469 and MX1041X261 cases

In this subsection, we will test two cases. The first case, the MX521X469 case, has $521 \times 469 \times 20 = 4.88$ million grid blocks, and all the blocks are active. The permeability and porosity are constant. It is a three-phase water injection problem and is simulated for 4000 days. It has 90 water injection wells and 110 production wells. The second case, the MX1041X261 case, has $1041 \times 261 \times 20 = 5.43$ million grid blocks, all active. The permeability and porosity on each vertical layer are constant. It is also a three-phase water injection problem and is simulated for 4000 days. There are more wells than the first case, which are 126 production wells and 100 water injection wells. More details about these two cases can be found

Table 6
MX521X469 case.

NP	Time steps	Newton	Linear	A.L.N.	Time (sec)	Scalability
64	1534	5656	23,711	4.2	55,820	–
128	1529	5467	21,925	4.0	30,691	0.91
256	1539	5752	22,682	3.9	16,932	0.82
512	1538	5692	23,569	4.1	9126	0.77
1024	1539	5727	23,152	4.0	5006	0.70

Table 7
MX1041X261 case.

NP	Time steps	Newton	Linear	A.L.N.	Time (sec)	Scalability
64	1243	4092	37,053	9.1	41,683	–
128	1241	4132	26,410	6.4	22,854	0.91
256	1235	4001	27,426	6.9	13,383	0.78
512	1240	4049	35,339	8.7	7414	0.70
1024	1241	4058	33,078	8.2	4478	0.58

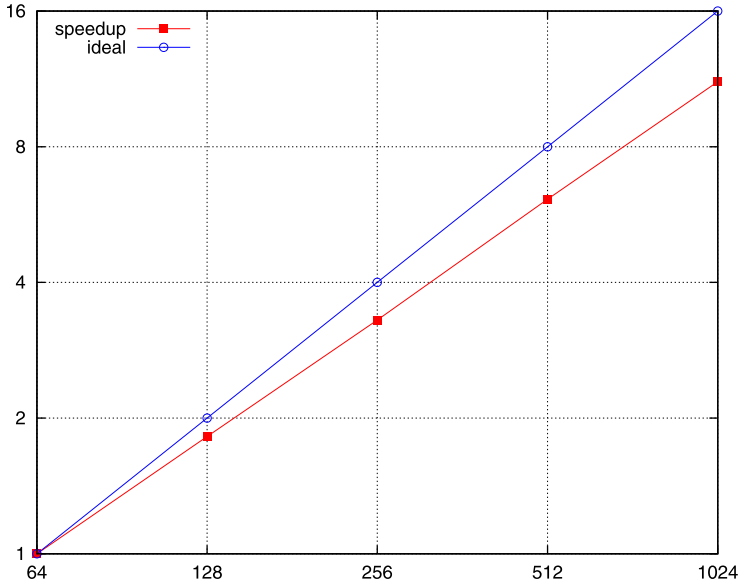

Fig. 5.6. Speedup of MX521X469 case.

in [40]. The efficiency and scalability of the three-stage preconditioner and the design of our simulator for cases with hundreds of wells are tested. We use from 64 to 1024 MPI processes.

From the results of the MX521X469 case and the MX1041X261 case, Tables 6 and 7, we can see that the number of time steps and Newton iterations are both robust. Since the permeability is isotropic, the linear systems resulted from these two cases are relatively easier to solve, and around 4 and 8 times linear iterations are needed by each Newton iteration for the MX521X469 case and the MX1041X261 case, respectively. The effectiveness of the three-stage preconditioner is not affected by the increasing of the wells. Encouraging parallel scalability is obtained when 1024 MPI processes are used, which can demonstrate the reasonable design of our parallel simulator. The speedup of the two cases can be seen in Figs. 5.6 and 5.7, respectively.

5.5. Refined SPE10 oil–water case

In this case, the original SPE10 two phase oil–water problem is refined into a problem with 140 millions of grid blocks. This case is used to test the capacity of our simulator to handle extremely large scale problems and re-test the three-stage preconditioner. For this extremely large scale problem, we start at using 256 MPI processes, and set the simulation period to 2 days.

As shown in Table 8, up to 2048 MPI processes are employed in this case, and excellent scalability is obtained on IMB Blue Gene/Q. Moreover, the speedup is slightly superlinear, see Fig. 5.8. The robustness of the three-stage preconditioner is verified again in this case, where the number of Newton iterations and linear iterations are fixed at around 210 and 5500 with different number of MPI processes, respectively.

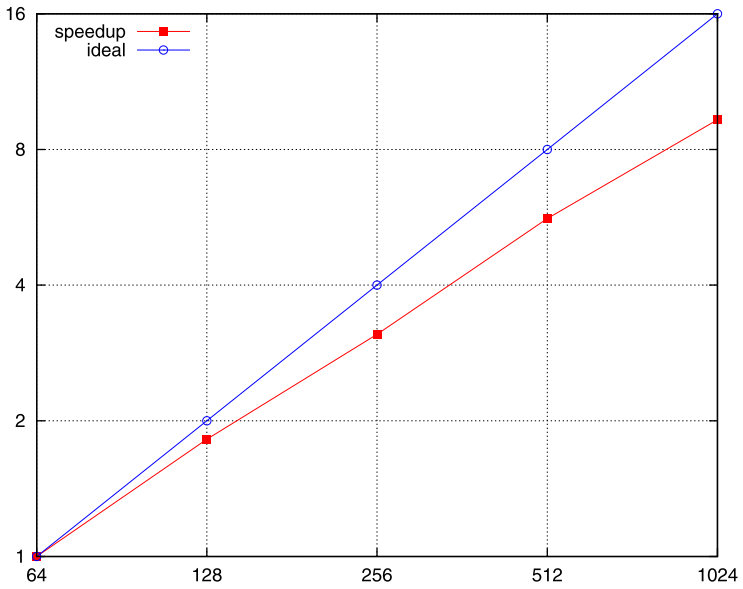

Fig. 5.7. Speedup of MX1041X261 case.

Table 8
Refined SPE10 oil–water case.

NP	Time steps	Newton	Linear	A.L.N.	Time (sec)	Scalability
256	36	225	5544	24.6	117,288	–
512	36	226	5724	25.3	57,643	1.02
1024	35	207	5446	26.3	27,370	1.07
2048	36	209	5530	26.4	14,275	1.03

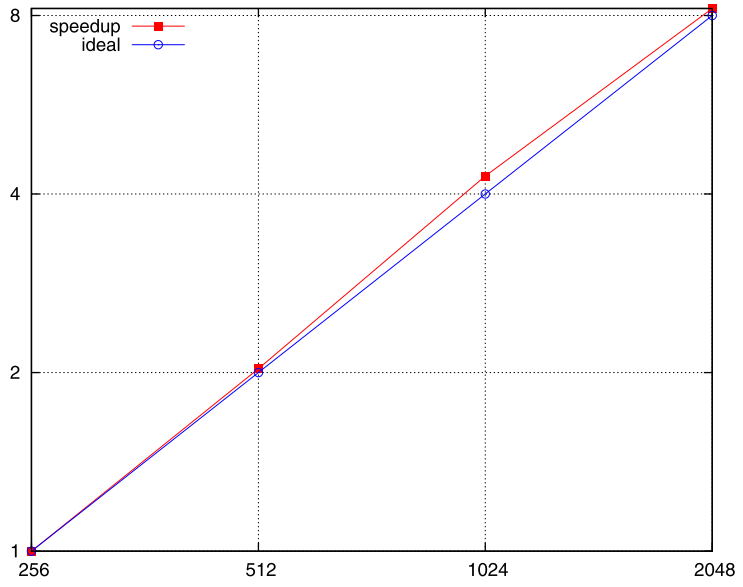

Fig. 5.8. Speedup of refined SPE10 oil–water case.

6. Conclusions

In this paper, we present our work on developing a parallel reservoir simulator. The numerical discretization scheme of the black oil model, an inexact Newton method, parallel linear solvers and advanced matrix pre-processing technique are studied. The implementation is based on PRSI, a general purpose platform for reservoir simulation. Its grid management

and well management are the basis of our simulator's parallel scalability, and its linear solver library is the assurance of our preconditioner's high efficiency. The inexact Newton method prevents the linear system from oversolved, which saves a lot of unnecessary computations. For the linear systems formed from the black oil model, we designed a new three-stage preconditioner. Our simulator has been validated by testing the SPE1 and SPE10-based problems. With the SPE10-based problem and the refined SPE10-based problem, we compared the performance of the RAS-ILU preconditioner, the CPR preconditioner, the FASP preconditioner and the three-stage preconditioner by detail, and the results show our preconditioner is more effective and more efficient than the others. The MX521X469 and MX1041X261 large scale problems with hundreds of wells are also tested, and the three-stage preconditioner, as well as the parallel simulator, have robust performance. In the last case, an oil–water case with 130 millions of grid blocks and highly heterogeneous permeability is tested to show our simulator's capacity of handling extremely large scale problems. From the parallel scalability tests of each case, we can see that the number of Newton iterations and linear iterations are stable as increasing the number of MPI processes. Encouraging parallel scalability is obtained, and the computing time is nearly halved when the MPI processes are doubled. All the numerical experiments demonstrate that our simulator is reliable, efficient, parallel scalable and capable of handling very large-scale problems.

Acknowledgements

The support of Department of Chemical and Petroleum Engineering, University of Calgary and Reservoir Simulation Research Group is gratefully acknowledged. The research is partly supported by NSERC IRCPJ 365863/AIEES 1741/Foundation CMG, AITF iCore, IBM Thomas J. Watson Research Center, and the Frank and Sarah Meyer FCMG Collaboration Centre for Visualization and Simulation. The research is also enabled in part by support provided by WestGrid (www.westgrid.ca), SciNet (www.scinethpc.ca) and Compute Canada Calcul Canada (www.computecanada.ca).

References

[1] Z. Chen, G. Huan, Y. Ma, Computational Methods for Multiphase Flows in Porous Media, vol. 2, SIAM, 2006.
[2] K. Stüben, A review of algebraic multigrid, J. Comput. Appl. Math. 128 (1) (2001) 281–309.
[3] J.R. Wallis, Incomplete Gaussian elimination as a preconditioning for generalized conjugate gradient acceleration, in: SPE Reservoir Simulation Symposium, 1983.
[4] G.A. Behie, P.A. Forsyth Jr., Incomplete factorization methods for fully implicit simulation of enhanced oil recovery, SIAM J. Sci. Stat. Comput. 5 (3) (1984) 541–561.
[5] J.R. Wallis, R.P. Kendall, T.E. Little, Constrained residual acceleration of conjugate residual methods, in: SPE Reservoir Simulation Symposium, 1985.
[6] H. Cao, H.A. Tchelepi, J.R. Wallis, H.E. Yardumian, Parallel scalable unstructured CPR-type linear solver for reservoir simulation, in: SPE Annual Technical Conference and Exhibition, 2005.
[7] 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 Numer. Math. 29 (4) (1989) 938–954.
[8] K. Li, A. Dogru, A. McDonald, A. Merchant, A. Al-Mulhem, S. Al-Ruwaili, N. Sobh, H. Al-Sunaidi, Improving the performance of mars simulator on cray-2 supercomputer, in: SPE Middle East Oil Show in Bahrain, 11–14 March 1995.
[9] J.A. Wheeler, R.A. Smith, Reservoir simulation on a hypercube, SPE 19804, in: The 64th Annual SPE Conference & Exhibition, San Antonio, October 1989.
[10] J.E. Killough, R. Bhogeswara, Simulation of compositional reservoir phenomena on a distributed-memory parallel computer, J. Pet. Technol. 43 (11) (1991) 1368–1374.
[11] J.M. Rutledge, D.R. Jones, W.H. Chen, E.Y. Chung, The use of massively parallel SIMD computer for reservoir simulation, SPE Paper 21213, in: The Eleventh SPE Symposium on Reservoir Simulation, Anaheim, 1991.
[12] T. Kaarstad, J. Froyen, P. Bjorstad, M. Espedal, Massively parallel reservoir simulator, in: SPE Reservoir Simulation Symposium, San Antonio, Texas, 1995.
[13] G. Shiralkar, R. Stephenson, W. Joubert, O. Lubeck, B. van Bloemen Waanders, A production quality distributed memory reservoir simulator, in: SPE Reservoir Simulation Symposium, 1997.
[14] J.E. Killough, D. Camilleri, B.L. Darlow, J.A. Foster, Parallel reservoir simulator based on local grid refinement, in: SPE Reservoir Simulation Symposium, Dallas, 1997.
[15] M. Parashar, J.A. Wheeler, G. Pope, K. Wang, P. Wang, A new generation EOS compositional reservoir simulator: Part II framework and multiprocessing, Paper SPE 37977, in: The SPE Reservoir Simulation Symposium, Dallas, 8–11 June 1997.
[16] A.H. Dogru, L.S.K. Fung, U. Middya, T. Al-Shaalan, J.A. Pita, A next-generation parallel reservoir simulator for giant reservoirs, in: SPE/EAGE Reservoir Characterization & Simulation Conference, 2009.
[17] H. Liu, S. Yu, Z. Chen, B. Hsieh, L. Shao, Parallel preconditioners for reservoir simulation on GPU, in: SPE Latin America and Caribbean Petroleum Engineering Conference, 2012.
[18] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, SIAM J. Sci. Comput. 21 (2) (1999) 792–797.
[19] Z. Chen, H. Liu, S. Yu, B. Hsieh, L. Shao, GPU-based parallel reservoir simulators, in: Proc. of 21st International Conference on Domain Decomposition Methods, France, 2012.
[20] T.M. Al-Shaalan, H.M. Klie, A.H. Dogru, M.F. Wheeler, Studies of robust two stage preconditioners for the solution of fully implicit multiphase flow problems, in: SPE Reservoir Simulation Symposium, 2009.
[21] X. Hu, W. Liu, G. Qin, J. Xu, C. Zhang, Development of a fast auxiliary subspace pre-conditioner for numerical reservoir simulators, in: SPE Reservoir Characterisation and Simulation Conference and Exhibition, 2011.
[22] C. Feng, S. Shu, J. Xu, C.-S. Zhang, A multi-stage preconditioner for the black oil model and its OpenMP implementation, in: Domain Decomposition Methods in Science and Engineering XXI, Springer International Publishing, 2014, pp. 141–153.
[23] S. Yu, H. Liu, Z. Chen, B. Hsieh, L. Shao, GPU-based parallel reservoir simulation for large-scale simulation problems, in: SPE Europec/EAGE Annual Conference, Copenhagen, Denmark, 2012.
[24] H. Liu, S. Yu, Z. Chen, Development of algebraic multigrid solvers using GPUs, in: SPE Reservoir Simulation Symposium, Houston, USA, February 2013.
[25] Z. Chen, H. Liu, S. Yu, B. Hsieh, L. Shao, Reservoir simulation on NVIDIA Tesla GPUs, in: The Eighth International Conference on Scientific Computing and Applications, University of Nevada, Las Vegas, April 2012.

[26] H. Klie, H. Sudan, R. Li, Y. Saad, Exploiting capabilities of many core platforms in reservoir simulation, in: SPE RSS Reservoir Simulation Symposium, 21–23 February 2011.
[27] R. Li, Y. Saad, GPU-accelerated preconditioned iterative linear solvers, Technical report umsi-2010-112, Minnesota Supercomputer Institute, Minneapolis, MN, 2010.
[28] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003.
[29] T. Chen, N. Gewecke, Z. Li, A. Rubiano, R. Shuttleworth, B. Yang, X. Zhong, Fast computational methods for reservoir flow models, University of Minnesota, Institute of Mathematics and its Applications, 2009.
[30] S. Lacroix, Y.V. Vassilevski, M.F. Wheeler, Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS), Numer. Linear Algebra Appl. 8 (8) (2001) 537–549.
[31] R. Scheichl, R. Masson, J. Wendebourg, Decoupling and block preconditioning for sedimentary basin simulations, Comput. Geosci. 7 (2003) 295–318.
[32] A. Elli, O.B. Widlund, Domain Decomposition Methods: Algorithms and Theory, vol. 3, Springer, Berlin, 2005.
[33] R.D. Falgout, U.M. Yang, hypre: a library of high performance preconditioners, in: Computational Science-ICCS 2002, Springer, Berlin, Heidelberg, 2002, pp. 632–641.
[34] G. Karypis, K. Schloegel, V. Kumar, Parmetis: parallel graph partitioning and sparse matrix ordering library, version 1.0, Technical report, Dept. of Computer Science, University of Minnesota, 1997.
[35] E. Boman, K. Devine, R. Heaphy, B. Hendrickson, V. Leung, L. Riesen, C. Vaughan, U. Catalyurek, D. Bozdag, W. Mitchell, J. Teresco, Zoltan v3: parallel partitioning, load balancing and data-management services, user's guide, Technical report SAND2007-4748W, Sandia National Laboratories, 2007.
[36] H. Liu, Dynamic load balancing on adaptive unstructured meshes, in: 10th IEEE International Conference on High Performance Computing and Communications, 2008.
[37] C. Loken, D. Gruner, L. Groer, R. Peltier, N. Bunn, M. Craig, T. Henriques, J. Dempsey, C.-H. Yu, J. Chen, et al., SciNet: lessons learned from building a power-efficient top-20 system and data centre, J. Phys. Conf. Ser. 256 (2010) 012026, IOP Publishing.
[38] A. Odeh, Comparison of solutions to a three-dimensional black-oil reservoir simulation problem, J. Pet. Technol. 33 (1) (1981) 13–25.
[39] M.A. Christie, M.J. Blunt, Tenth SPE comparative solution project: a comparison of upscaling techniques, SPE Reserv. Eval. Eng. 4 (4) (2001) 308–317.
[40] G.L. Brown, D.A. Collins, Z. Chen, Efficient preconditioning for algebraic multigrid and red–black ordering in adaptive-implicit black-oil simulations, in: SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, 2015.