SPE 51885

A Fully Implicit Parallel EOS Compositional Simulator for Large Scale Reservoir Simulation.

P. Wang, S. Balay[1], K.Sepehrnoori, J. Wheeler, J. Abate, B. Smith[1], G.A. Pope. The University of Texas at Austin. [1]Argonne National Laboratory

This paper was prepared for presentation at the 1999 SPE 15th Reservoir Simulation Symposium, Feb. 14-17, Houston Texas.

Abstract

A fully implicit parallel equation-of-state (EOS) compositional simulator for large-scale reservoir simulation is presented. This simulator is developed under the framework named IPARS (Integrated Parallel Accurate Reservoir Simulator) and is constructed using a Newton-type formulation. The Peng-Robinson EOS is used for the hydrocarbon phase behavior calculations. The linear solvers from the PETSc package (Portable Extensible Toolkit for Scientific Computation) are used for the solution of the underlying linear equations. The framework provides input/output, table lookups, Fortran array memory allocation, domain decomposition, and message passing between processors for updating physical properties in mass-balance equations in overlapping regions. PETSc handles communications between processors needed for the linear solver.

Many test runs were performed with up to four million gridblocks for a dry-gas injection process on an IBM SP machine and half a million gridblocks on a cluster of 16 PCs. Results indicate that the scalability of the simulator is very good. The linear solver takes around half of the total computational time for homogeneous reservoirs. For layered heterogeneous reservoirs, the linear solver took a larger fraction of the total computational time as the permeability contrast increased. The time for the communication between processors for updating the flow equations is insignificant.

The PC cluster is roughly a factor of two slower than the SP for parallel runs, which is very encouraging. This factor is strongly related to the hardware configuration of the computers, which is detailed in the paper.

Introduction

The importance and necessity for performing reservoir simulation studies on parallel computers have been well addressed in many recent publications[1-4]. Much effort has been invested in this area, which has led to significant progress in the last few years.

The overall objective of this research is the development of a new-generation framework and simulator suitable for both massively parallel processors and clusters of PCs. This research was initiated in 1995 as a joint effort by UT and Argonne National Laboratory. The product of this research is intended for use as a vehicle for studying ideas and algorithms aimed at issues associated with parallel processing of reservoir simulations.

The simulator formulation and the solution procedure have been described in our earlier paper[5]. In this paper, we focus on the following:

- the framework approach we used to develop and implement an equation-of-state (EOS) compositional parallel reservoir simulator
- the outcome of our research for large-scale reservoir simulations with millions of gridblocks on both IBM SP machines and a cluster of PCs.

Our experience has shown that the framework approach works well for processing reservoir simulations on parallel computers. The complicated issues associated with parallel processing are as much as possible handled by the framework, and the model developers solely focus on the formulation and physical properties of the simulator and then use the functions provided by the framework for processing the computations in parallel.

Description of the Governing Equations

Multicomponent and multiphase flow in a porous medium can be described using three different types of equations when temperature is constant:

- partial differential, component-mass balances describing component flow, in which Darcy's law is used to govern the transport of phases from one cell to another;
- phase equilibrium equations dealing with equilibrium component mass transfer between phases; and

- equations constraining phase saturation and component concentrations.

Neglecting mutual solubility between water and hydrocarbon phases, for a system consisting of n_c hydrocarbon components and n_p fluid phases (excluding the aqueous phase), the above three types of equations may be mathematically expressed for a control volume as follows (the dispersion term is ignored for simplicity):

Component material balance in terms of moles per unit time

$$V_b \frac{\partial N_i}{\partial t} - V_b \nabla \bullet \sum_{j=1}^{n_p} \frac{k k_{rj} x_j}{m_j} x_{ij} (\nabla P_j - g_j \nabla D) - q_i = 0 \qquad (1)$$

for $i = 1,2,...,n_c$. For water, x_{ij}=1 in the above equation. The pressure difference between phases is evaluated using the capillary pressure. This equation also holds for water by inserting the properties of the aqueous phase.

Phase-equilibrium relationship.

The first partial derivative of the total Gibbs free energy with respect to the independent variables gives equality of component fugacities among all phases:

$$f_i^{oil} - f_i^{gas} = 0 \qquad (2)$$

for n_c variables. The following equations are used in the solution of Eq. (2):

$$\sum_{i=1}^{n_c} \frac{z_i (K_i - 1)}{1 + v(K_i - 1)} = 0, \qquad (2a)$$

$$\sum_{i=1}^{n_c} x_{ij} - 1 = 0. \qquad (2b)$$

Volume constraints.

The pore volume in each cell must be filled completely by the total fluid volume:

$$V_b \sum_{i=1}^{n_c} N_i \sum_{j=1}^{n_p} L_j \bar{v}_j - V_p = 0 \qquad (3)$$

The independent variables used in the simulator are $P, N_1, \cdots, N_{nc}, \ln K_1, \cdots, \ln K_{nc}, N_w$.

Solution Procedure

We discretized the component-mass balance equation, Eq. (1), on a rectangular grid using the one-point upstream weighting scheme for the transmissibility terms, which leads to a set of nonlinear equations that is solved by the Newton method. The discretized component mass-balance equations along with the fugacity and volume constraints were then linearized in terms of the independent variables. Therefore, the number of linear equations for a reservoir represented by n_b gridblocks is $(2n_c + 2)n_b$, which are solved simultaneously by one of the linear solvers of PETSc (Portable Extensible Toolkit for Scientific Computation), which is described in detail later.

During each update of the independent variables, a phase-stability test is performed for each gridblock to check if the fluid changes its phase number. For fluid changing from one hydrocarbon phase to two hydrocarbon phases, the flash calculation is done to determine composition, amount and density of each separate phase for initializing the iteration for the next Newton step. An option is also available to perform the phase-stability test for gridblocks adjacent to one having both gas and oil. The gridblocks containing wells are always checked for phase-number change.

Because the linearized system includes both the flash and volume constraint equations, the gridblocks representing only gas or oil have their corresponding off-diagonal elements in the Jacobian set to zero, and those on the diagonal are set to one. The corresponding values in the right-hand side vector are set to zero. The detailed description of the simulator and the fluid-related calculations can be found elsewhere[5].

The Framework

In this research, we employed a framework approach to handle the complicated tasks associated with parallel processing. The purpose is to separate as much as possible the physical model development from parallel processing so that code implementation in both models and parallel processing can proceed at the same time by different team members.

Based on this design idea, we have developed a framework named the Integrated Parallel Accurate Reservoir Simulator (IPARS)[6]. Under this framework, several physical models have been developed, such as two-phase water-oil and black-oil models in both IMPES and fully implicit formulations that are implemented in conjunction with the multiblock algorithm[7-8]. The EOS compositional model is only one of the models running under the framework.

The tasks handled by the framework in conjunction with the EOS compositional model include:

- I*nput/Output*. IPARS allows each processor to read in the entire input but process only the portion for which a processor is responsible. The output is collected by the masterprocessor (processor 0) and then sent to the proper files.
- *Domain splitting*. The original reservoir simulation domain is divided vertically into several subdomains equal to number of processors required by the run. The computations associated with each of these subdomains are assigned to each processor.
- *Memory allocation*. The framework allocates memory for grid-element arrays in the format of I, J, and K being the first three indices representing the three coordinate directions. The indices identifying components and phases are positioned as the fourth and fifth dimension. A physical model can create as many grid-element arrays

as needed. The model developers need to provide the variable name, ID in the memory, and size of extra dimensions. These variable IDs are assigned by the framework using C language when creating the variables and stored in a common block for use in different routines.

- *Message passing between processors.* The message passing interface (MPI) is used in the framework to handle necessary message sending/receiving between processors. In the physical models, this task is also carried out through a function call and is performed when evaluating the residuals of the component mass-balance equations, averaging some physical properties such as pressure and saturation, and collecting well data for output. The update calls require the model developers to provide the name and ID in the memory for the properties that need to be updated in the overlapping regions of all of the processors. Note that the necessary communications during the solution of the linear equations are handled by PETSc.
- *Others.* Some common calculations needed by almost all of the physical models are also handled by the framework. These include:
  1. identification of the overlapping regions of all of the processors.
  2. the constant portion of transmissibility at the interface of two adjacent gridblocks.
  3. table lookups for phase relative permeability and capillary pressure and their derivatives with respect to phase saturation.
  4. well locations and productivity indices.
  5. preprocessing, which performs computer platform and simulation run dependent tasks. These include dimensioning Fortran arrays for non-grid-element variables, estimating the size of message in bytes to be needed for passing between processors, and some dimensions for well specifications.

Implementation of the EOS Compositional Model

The framework IPARS provides hooks for implementing a specific physical model. Such hooks bridge the model with the framework.

Several major hooks designed by the framework for the EOS compositional model are:

- *xisdat* - input initial scalar data. These include physical properties of each component, default number of iterations and convergence tolerances for a variety of calculations, output flags, and operation-specific flags. No grid-element arrays can be referenced in this routine.
- *xarray* - allocate memory for all of the grid-element arrays.
- *xiadat* – complete input and print it out to a file.
- *xivdat* – perform model initialization before time iteration. The PETSc linear solver is also initialized.
- *xstep* – perform all of the calculations over a timestep.
- *xquit* – exit simulation when it meets the maximum time, production limits, or an error occurs.

These routines are called the executive routines, and any communications between processors for the compositional model are performed in these routines. The model developers need to insert into each of the above routines the codes that perform the corresponding tasks. The well calculations are contained in the routine *xstep*.

There is no argument attached to these calls. Those variables associated with grids are passed into these routines through pointers that are stored in a common block. The calculations associated with these executive routines need to be performed in the work routines that are called from the executive routines. The grid dimensions and variables are passed into these work routines through a C routine (handled by the framework).

The most important routine for the EOS compositional model is *xstep*. It handles the following operations at a given timestep:

1. compute the dependent variables based on the independent variables
2. update properties in the overlapping regions of all of the processors (communications between processors)
3. evaluate the residuals for all of the governing equations. For the component mass-balance equations, it involves a summation of the residuals from each of the processors. The function for doing so is provided by the framework
4. select the maximum residuals for each type of the governing equations among all of the processors
5. check if a convergence is reached on the masterprocessor (processor 0) and broadcast the outcome of the check to other processors
6. if the tolerances are met, output user-specified properties at this current time level and then return to the framework for timestepping calculation for the next timestep
7. if the tolerances are not met, evaluate the element of the Jacobian and then call PETSc linear solver
8. update the independent variables and go back to step 2 for the Newton iteration.

PETSc Solvers

PETSc 2.0[9] is a large suite of parallel general-purpose object-oriented time-steppers, nonlinear and linear solvers[10] for the scalable solution of partial-differential equations discretized using implicit and semi-implicit methods. PETSc is implemented in C, and is usable from C, Fortran, and C++. It uses MPI[11] for communication across processors. PETSc has been used for a wide variety of applications, including computational fluid dynamics, structural dynamics, materials modeling, and econometrics. Many of the solvers are appropriate for problems discretized using either structured grids or unstructured grids. The EOS compositional simulator uses the linear solver component of PETSc to solve the linearized Newton system of equations and uses the parallel data formats provided by PETSc to store the Jacobian

and the vectors. In this particular application, the linear systems arise from a structured grid.

The linear solver component of PETSc provides a unified interface to various Krylov methods, such as conjugate gradient (CG), generalized minimal residual (GMRES), biconjugate gradient, etc. It also provides a unified interface to various parallel preconditioners such as Jacobi, block preconditioners like block Jacobi, domain decomposition preconditioners like additive Schwartz. The application uses this common interface and thus can choose the appropriate Krylov method and the preconditioner at runtime. The simulator uses the biconjugate gradient stabilized approach as the Krylov method and block Jacobi preconditioner, with point-block incomplete factorization (ILU) on the subdomain blocks. Here, point-block refers to treating all the variables associated with a single gridblock as a single unit. The number of subdomain blocks for block Jacobi is chosen to match the number of processors used, so that each processor gets a complete subdomain of the problem and does a single local incomplete factorization on the Jacobian corresponding to this subdomain.

PETSc provides various sparse matrix storage formats including basic sparse storage, point-block sparse storage, point-block diagonal storage, etc. All the matrix storage formats have a uniform interface to the matrix operations. The solvers use this interface to access the matrix operations. This enables the solvers to work with any of the PETSc matrix storage formats. Also, internally, each storage format is associated with its own implementation of the matrix operations, which take into account the corresponding storage format. In this way, the individual routines are tuned to provide the best performance with the corresponding matrix format. For example, the point-block storage format makes use of the block structure and stores each point-block elements contiguously. Its corresponding matrix operations, including the matrix vector product, factorization and triangular solve, take advantage of this block structure. Also, each point-block size has a routine, which is written for this specific block size, thus giving the best possible performance on cache-based RISC machines.

The EOS compositional model for the three phase flow, generates $2n_c + 2$ equations per gridblock. This causes the Jacobian to have a point-block structure. Therefore, the point-block sparse storage format is used to store this matrix. These equations do not result in complete coupling of all the variables across gridblocks. This causes the Jacobian matrix to have some $n_c + 1$ point-block locations with zero values. Therefore, a block size of $n_c + 1$ is chosen for this matrix type, which eliminates the need to store the $(n_c + 1)(n_c + 1)$ blocks with zero values.

Table 1 demonstrates the improvement in performance of the matrix routines when the point-block sparse matrix storage format, with its associated matrix operations, is used, compared to the basic version. The runs were done on a single processor using a sample problem with 2,800 nodes, and four equations per node. The table shows the performance in Megaflops; the timings are done using wall-clock time.

For example on the IBM Power 2 Super Chip(P2SC), one sees the block format essentially doubles the performance, hence reducing the computation time by half.

Table 1. Megaflop rates of the matrix routines on different computers.

Machine Type	IBM SP		SGI Origin 2000		PC with NT	
Processor	P2SC		MIPS R10000		Pentium II	
Clock Speed	120 MHz		250 MHz		400 MHz	
	Basic	Blocked	Basic	Blocked	Basic	Blocked
MatMult	72	151	40	68	49	64
MatSolve	51	114	34	62	35	68

Simulator Performance

Many test runs have been performed using this fully implicit EOS compositional simulator on both IBM SP machines and a cluster of 16-node PCs. In this section, we describe the hardware configuration of the SP at Maui High Performance Computing Center (MHPCC), on which we performed the runs, and a PC cluster.

IBM SP Specifications at MHPCC

We ran our simulator on a 128-node SP machine with IBM Power2 Super Chip (P2SC) processors with a clock speed of 160 MHz. The communications between processors are through the SP high-performance switch. Each node has 512 MB of memory.

The standard IBM compilers *xlf* and *xlc* were used to generate the code executables with optimization level *O3*. IBM's implementation of MPI was used for the communications.

Description of the Test Cases

Several cases with different numbers of components and well patterns were tested. Each case was run several times with different numbers of gridblocks in order to evaluate the simulator scalability. All of the cases simulate a dry-gas injection process, and use the same porosity and relative permeability unless otherwise noted.

Case 1.

The reservoir fluid is described using a three-component Peng-Robinson EOS. The reservoir is homogeneous and contains two wells, an injector located in one corner and a producer installed in the opposite corner. Initial conditions and production scheme were specified so that gas/oil/water are all present during the entire simulations but water is immobile. The sizes of each gridblock in x, y, and z

directions are 100 ft, 100 ft, and 50 ft, respectively. The simulation was for 1000 days. The linear solver options used are described in the previous section.

Figure 1 shows the speedup for gridblocks up to 1,000,000. The curves are normalized against the timing from the run with the lowest number of processors. It can be seen that the simulator exhibits very good scalability. For the problems with a small number of gridblocks, the performance was reduced somewhat because the amount of computation on each of the processors was insufficient. Figure 2 demonstrates the speedup for the runs where the ratio of the number of gridblocks to number of nodes (Rgn) is constant; in this case, Rgn = 20,480, as number of processors increases, namely the problem size increases as number of processors increases to maintain computational intensity on each processor. Figure 2 indicates that good scalability can be achieved if each processor has enough to work on.

Figure 3 shows the breakdown of the computational time for different tasks for the 1-million-gridblock case. As can be seen, the linear solver takes the largest portion of the total computational time; the next is the Jacobian update. Note that the time for MPI in this figure is one corresponding to the communication required for evaluating the residuals of the component mass-balance equations, and clearly it is not significant. The time on the MPI involved in the linear solver is excluded from this count because it is handled by PETSc and included in the solver time.

To update physical properties in the overlapping regions, one can perform all of the calculations in these regions without the need for communicating with other processors (except the need for collecting the residuals from all processors using the masterprocessor). Such an approach is at the expense of intensifying the computations on each processor. As indicated in Figure 3, in which the calculations in the overlapping regions were not done but updated through MPI, the communication time is less than 1%. So using MPI to update the physical properties in these regions was a more efficient way and hence has been used in all of the test cases.

For cases shown in Figures 1 through 3, the simulation domains were split vertically in one direction from one side of the reservoir to the other because the domains were setup for a long reservoir. We also performed the timings for runs where the two-dimensional domain splitting was used to divide the simulation domains among assigned processors. Figure 4 shows one of such tests that has a grid of 72 x 72 x 20 (103,680 gridblocks). It appears that the simulator performs well, although the scalability is not as good as that achieved from doing the 1D domain splitting. The breakdown of the computational time on different tasks for runs whose domains were divided vertically in two directions was similar to those in Figure 3. No direct comparison between these two splittings was made because the framework, for a given case, automatically divides the domain in one of the two ways, but not both.

The runs described above used a homogeneous permeability field, where $k_x = k_y = 50$, and $k_z = 20$. We also evaluated the simulator performance for layered permeability fields with contrast up to 250 to 1. It was observed that more time was spent on the linear solver as the heterogeneity increased. One such result is given in Figure 5, in which the permeability contrast is 200 to 1. The CPU for the solver increased from 50% of the total computational time for the homogeneous run to 80% for the layered permeability run, and the total computational time was increased proportionally.

For runs made in this case, we have used up to 4.032 million gridblocks on the 128 processors, which drove the linear solver to solve a system with 32.256-million unknowns simultaneously, because a reduction in the size of the Jacobian through a row elimination using Eq.(2) was not done in these runs. For this largest run we made, it took less than half an hour to make seven timestep iterations that involved solving the 32.256-million unknowns 21 times for a 200-day simulation.

Case 2.

In this case, we used a six-component PR EOS to represent the reservoir fluid and a five-spot well pattern for the production scheme; otherwise run specifications were the same as those in Case 1. Figure 6 shows the speedup of one of the runs that used 64 x 64 x 8 grids for five years of production. It indicates that the simulator scalability remains very good.

As can be seen, in some of the runs, the number of gridblocks is not very large because we intended to compare our simulator performance between different computer platforms, especially a cluster of PCs. For the cluster of PCs that we used, the hardware and software are briefly described as follows:

- 16 nodes, 300 MHz Intel Pentium II processors, each with 384 MB memory, 66 MHz internal bus;
- Connected by switched 100 Mbps Ethernet (use Intel EtherExpress Pro/100 Mbps network cards and Intel 510T 10/100 Mbps switch); and
- RedHat Linux 5.0, GNU C/C++ compilers and Portland Group Fortran 77/90 compilers, MPICH, an implementation of MPI from Argonne National Laboratory, for parallel communication.

Figure 6 shows the speedup on the cluster of PCs, which is not quite as good as that for the SP when the number of processors increases from 8 through 16. However, by putting enough gridblocks on each processor to keep the number of gridblocks per processor constant (Rgn in Figure 6), we observed almost identical speedup on both the SP and the cluster of PCs. This case would be more realistic because we would try first to allocate as many gridblocks as possible to each processor to maximize the use of its resources. Figure 7 shows the CPU distribution for different tasks for this case on the PCs as a function of number of processors. The solver time increased as we used more processors. There were no significant load-balancing problems.

The largest run we made on the 16-node PC cluster was on 512,000 gridblocks using three-component PR EOS and two wells.

Case 3.

To further evaluate the performance of the simulator on both the SP and the 16-node PC cluster , we set up a case that is close to a full-field scale (initial oil in place is on the order of 10^9STB) with homogeneous and layered anisotropic permeability and with a relatively realistic production scheme. The dry gas with three hydrocarbon components started injection through four injectors into the reservoir after a one-year primary production through nine producers. The injectors were installed only in the top nine layers and the producers penetrated the bottom nine layers. These wells are in operation during the entire simulation period of 10 years. Table 2 lists some basic specifications for this run.

Table 2. Basic specifications for the 207,360-gridblock and 13-well run.

Reservoir size (ft)	9,000 x 12,800 x 900
Grid	90 x 128 x 18
Permeability (md)	kx=ky=50, kz=30
Porosity	0.20
Residual water saturation	0.20

For the run with the layered anisotropic permeability, the largest and smallest permeabilities are 2,500 md and 5 md, respectively. The largest permeability contrast between two adjacent layers is 250 to 1.

We ran this case on the 16 processors. The simulation domain was divided vertically in one dimension along the y direction, which makes only five processors contain wells, Processors 0, 7, and 15 contain three producers per node, and processors 3 and 11 have two injectors per node. For the homogeneous case, the simulator took close to seven hours on the 16-node PC cluster and around four hours on the SP at MHPCC to complete the run. Figure 8 illustrates the breakdown of the computational time for the two most time-consuming tasks. It appears that the time for the linear solver drops significantly when running on the SP compared with PCs..

The run for the heterogeneous case had different computational loads on different processors because of wells and different numbers of phases in the different regions. To evaluate whether the load imbalance is significant for this heterogeneous case, we timed the minimum and maximum time spent among the 16 processors for some major computational tasks including solver, Jacobian update, dependent-variable updates, message passing, and the well calculations, which are shown in Figure 9 (except for the MPI and well-calculation times since their portions are too small to show on the chart). The figure clearly indicates that, for given tasks, different processors took different times. For the Jacobian update, for example, the minimum time for this task on a processor was 15 minutes less than the maximum time; for the linear solver, a difference of 10 minutes was observed. For the well calculations, we noticed a factor of two difference between the minimum and maximum times. Fortunately, these differences did not introduce any significant load-balancing problem for this run. All of the processors completed their jobs within a range of two minutes, which is 0.5% of the total computational time. The run took 11 hours and 15 minutes on the cluster.

For all of the runs tested, we observed the PC cluster is roughly a factor of 2 slower than the SP at MHPCC, which is very encouraging. The detailed analysis of the simulator performance on clusters of PCs compared to IBM SP machines is given in a separate paper[12].

Summary and Conclusion

The parallel EOS compositional simulator developed in this work is intended to serve as a vehicle for further studying ideas and algorithms associated with parallel processing of reservoir simulations. Our experience and simulation results show the follwoing:

1. The framework approach for developing parallel simulators works well; it can separate, to a significant degree, the framework from the physical models without loss of simulator efficiency. Such a separation enables the framework and physical model to be developed and implemented simultaneously.
2. The simulator exhibits very good scalability on both IBM SP machines and a cluster of PCs. Cost of the massage sending/receiving between processors for physical properties in the overlapping regions is insignificant.
3. The linear solvers scale very well with the number of processors. For the model problems they take roughly fifty percent of the computational time, this becomes higher for heterogeneous reservoirs.
4. A cluster of PCs with 16 nodes can be used effectively for large-scale reservoir simulation studies.

Future Work

Our research is still ongoing and simulator enhancement and refinement will be continuing. In the future, we plan to focus on the following:

1. Enhance fully implicit EOS compositional model with a black-oil option and other new features.
2. Improve the performance of the PETSc linear solvers with particular attention to
   a. optimizing the floating point performance on Intel Pentium II processors, and
   b. improving the PETSc preconditioner performance.
3. Update the framework to incorporate use of automatic differentiation tools for generation of the Jacobian matrix.
4. Generalize the EOS compositional model to include thermal and chemical options.

Acknowledgement
This research has been supported by the U.S. Department of Energy under the Natural Gas and Oil Technology Partnership Program, contract W-31-109ENG-38. We would also like to thank Maui High Performance Computing Center for the use of their IBM SP, Intel for donating the PCs and networking hardware, and Robert Schneider of our department for setting up the cluster.

Nomenclature

D	depth from a reference datum plane
f_i	residual of Eq. 2
K_i	equilibrium ratio for component *i*, y_i / x_i
k	formation permeability
k_{rj}	relative permeability for phase *j*
L_j	ratio of moles in phase *j* to the total number of moles in the mixture
N_i	moles of component *i* per unit bulk volume
n_b	number of cells
n_c	number of components
n_p	number of phases
n_w	moles of water
P_j	pressure of phase *j*
q_i	molar injection (positive) or production (negative) rate for component i
t	time
V_b	bulk volume for a cell
V_p	pore volume for a cell
v	gas-phase molar fraction in absence of water
v_j	molar volume of phase *j*
x_{ij}	mole fraction of component *i* in phase *j*
z_i	overall mole fraction of component *i*

Greek Symbols

g_j	gravity term for phase *j*, defined as $r_j g$
∇	gradient operator
$\vec{\nabla} \bullet$	divergence operator
x_j	molar density of phase *j*
m_j	viscosity of phase *j*
f	porosity

References

1. Shiralkar, G.S., Stephenson, R.E., Joubert, W., Lubeck, L., and Waanders, B.v.B. :"Falcon: A Production Quality Distributed Memory Reservoir Simulator," SPE Res. Eval. Eng., Oct. 1998.
2. Chien, M.C.H., Techelepi, H.A., Yardumian, H.E., and Chen, W.H.: "A Scalable Parallel Multi-Purpose Reservoir Simulator," paper SPE 37976 presented at the 1997 SPE Reservoir Simulation Symposium, Dallas, TX (June 8-11, 1997).
3. Parashar, M., Wheeler, J.A, Pope, G.A., and Wang, P.: "A New Generation EOS Compositional Reservoir Simulator: Part II – Framework and Multiprocessing," paper SPE 37977 presented at the 1997 SPE Reservoir Simulation Symposium, Dallas, TX (June 8-11, 1997).
4. Ghori, S.G., Wang, C., Lim, M.T., Pope, G.A., Sepehrnoori, K., and Wheeler, M.F.: "Compositional Reservoir Simulation on CM-5 and KSR1 Parallel Machines," paper SPE 29140 presented at the 13th SPE Symposium on Reservoir Simulation, San Antonio, TX (1995).
5. Wang, P., Yotov, I., Wheeler, M.F., Arbogast, T., Dawson, C., Parashar, M and Sepehrnoori, K.: "A New Generation EOS Compositional Reservoir Simulator: Part I – Formulation and Discretization," paper SPE 37979 presented at the 1997 SPE Reservoir Simulation Symposium, Dallas, TX (June 8-11, 1997).
6. Wheeler, J.: "Integrated Parallel Accurate reservoir Simulator (IPARS)," The 8th Annual Industrial Affiliates Meeting, Center for Subsurface Modeling, The University of Texas at Austin (Oct. 27-28, 1998).
7. Yotov, I. "Multigrid on the Interface for Mixed Finite Element Methods on Non-Matching Multiblock Grids," presented at the 10th International Conference on Domain Decomposition Methods, Boulder, Colorado (Aug. 1997).
8. Wheeler, M.F. "Conservative Discretization for Modeling Subsurface and Surface Flows," presented at Fourth SIAM Conference on Mathematical and Computational Issues in the Geosciences, Albuquerque, New Mexico (June 1997).
9. Balay, S., Gropp, W.D., Curfman McInnes, L, and Smith, B.: "PETSc 2.0 Users Manual," Argonne National Laboratory, ANL-95/11 - Revision 2.0.22 (April, 1998).
10. Balay, S., Gropp, W., McInnes, L.C. and Smith, B.: "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries," *Modern Software Tools in Scientific Computing*, Argonne National Laboratory (Oct. 1997).
11. The MPI Forum: "*MPI:* A Message-Passing Interface Standard," International J. Supercomputing Applications, Vol. 8, No. ¾ (1997).
12. Abate, J, Wang, P, and Sepehrnoori, K., G.A.: "Parallel Compositional Reservoir Simulation on a Cluster of PCs," submitted to the journal of Communications in Numerical Methods in Engineering, 1998.

Figure 1. Speedup for Case 1 on the IBM SP at MHPCC

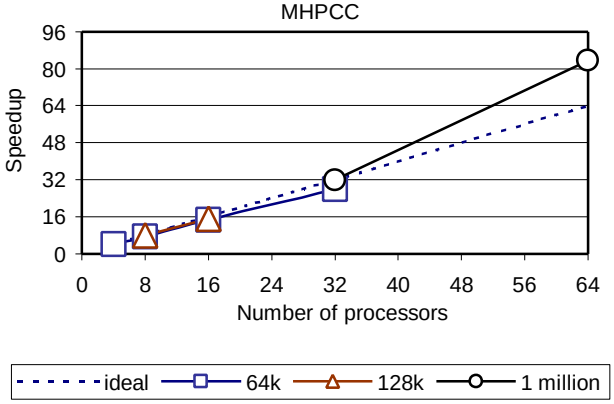

Figure 2. The speedup for Case 1 (Rgn=20,480)

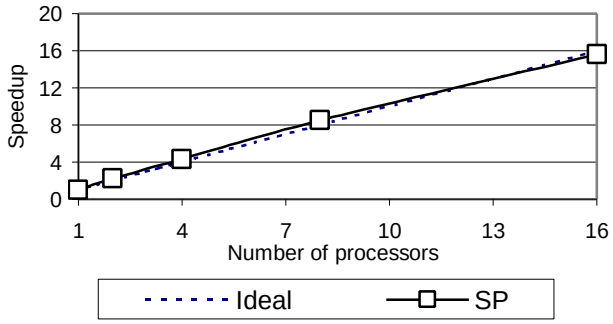

Figure 3. CPU Breakdow n for 1 million cells run.

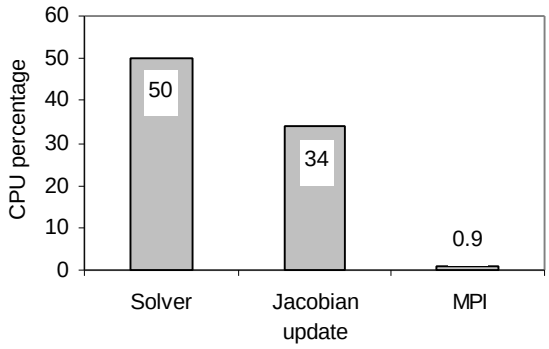

Figure 4. The speedup from 2D domain splitting for Case 1 (72 x 72 x 20 gridblocks).

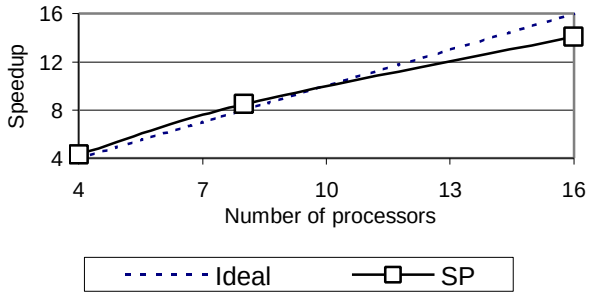

Figure 5. CPU distributions in 3 major tasks for the 512,000 cells run.

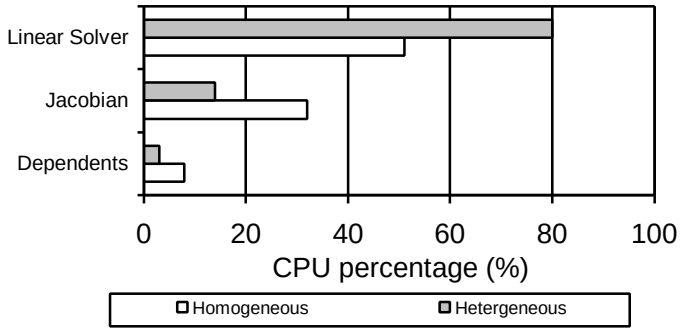

Figure 6. Speedup for a run with 5-spot well partern on both SP and PCs (64x64x8 gridblocks)

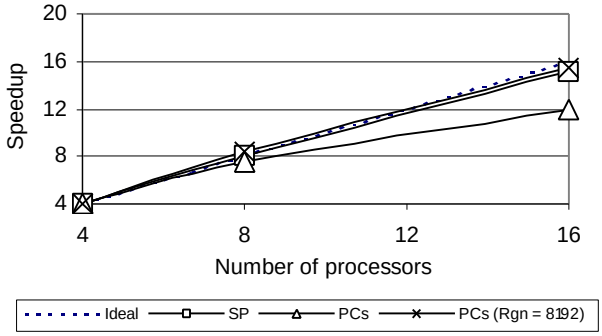

Figure 7. CPU distribution versus number processors

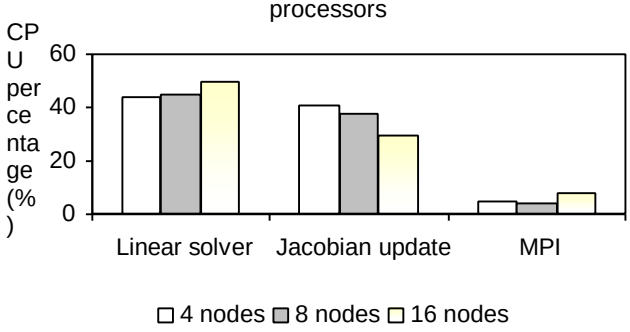

Figure 8. CPU breakdown for the run with 207,360 cells and 13 wells.

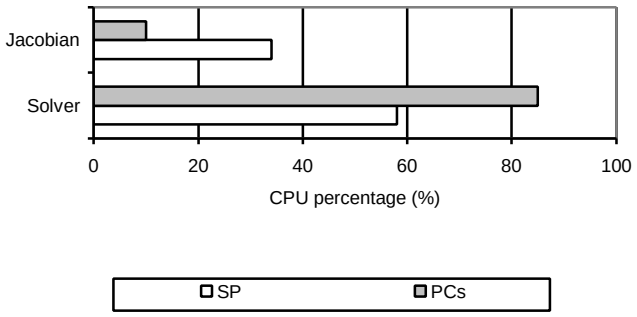

Figure 9. The max and min CPUs for the run with 207,360 cells and 13 wells on the 16 nodes.