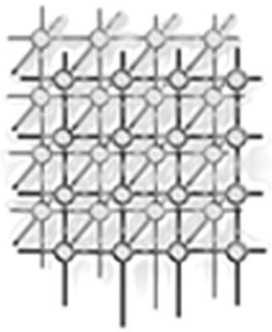
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2005; **17**:1441–1467
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.898



A simulation and data analysis system for large-scale, data-driven oil reservoir simulation studies

Tahsin Kurc[1,*,†], Umit Catalyurek[1], Xi Zhang[1], Joel Saltz[1], Ryan Martino[2], Mary Wheeler[2], Małgorzata Peszyńska[3], Alan Sussman[4], Christian Hansen[4], Mrinal Sen[5], Roustam Seifoullaev[5], Paul Stoffa[5], Carlos Torres-Verdin[6] and Manish Parashar[7]

[1]*Biomedical Informatics Department, The Ohio State University, 3184 Graves Hall, 333 West 10th Avenue, Columbus, OH 43210, U.S.A.*
[2]*Center for Subsurface Modeling, Institute for Computational Engineering and Sciences, The University of Texas at Austin, 201 East 24th Street, ACE 5.324, Campus Mail C0200, Austin, TX 78712, U.S.A.*
[3]*Department of Mathematics, Kidder Hall 368, Oregon State University, Corvallis, OR 97331-4605, U.S.A.*
[4]*Department of Computer Science, University of Maryland, A.V. Williams Building, College Park, MD 20742, U.S.A.*
[5]*Institute for Geophysics, The University of Texas at Austin, 4412 Spicewood Springs Road, Building 600, Austin, TX 78758-8500, U.S.A.*
[6]*Department of Petroleum and Geosystems Engineering, The University of Texas at Austin, 1 University Station, CO300, Austin, TX 78712-0228, U.S.A.*
[7]*Department of Electrical and Computer Engineering, Rutgers, State University of New Jersey, 94 Brett Road, Piscataway, NJ 08854-8058, U.S.A.*

SUMMARY

The main goal of oil reservoir management is to provide more efficient, cost-effective and environmentally safer production of oil from reservoirs. Numerical simulations can aid in the design and implementation of optimal production strategies. However, traditional simulation-based approaches to optimizing reservoir management are rapidly overwhelmed by data volume when large numbers of realizations are sought using detailed geologic descriptions. In this paper, we describe a software architecture to facilitate large-scale simulation studies, involving ensembles of long-running simulations and analysis of vast volumes of output data. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: oil reservoir simulation; Grid computing; data intensive computing; data management

*Correspondence to: Tahsin Kurc, Biomedical Informatics Department, The Ohio State University, 3184 Graves Hall, 333 West 10th Avenue, Columbus, OH 43210, U.S.A.
†E-mail: kurc@bmi.osu.edu

Received 20 April 2003
Revised 5 January 2004
Accepted 17 February 2004

1. INTRODUCTION

The main goal of oil reservoir management is to provide more efficient, cost-effective and environmentally safer production of oil from reservoirs. Implementing effective oil and gas production requires optimized placement of wells in a reservoir. A production management environment (see Figure 1) involves accurate characterization of the reservoir and management strategies that involve interactions between data, reservoir models, and human evaluation. In this setting, a good understanding and monitoring of changing fluid and rock properties in the reservoir is necessary for the effective design and evaluation of management strategies. Despite technological advances, however, operators still have at best a partial knowledge of critical parameters, such as rock permeability, which govern production rates; as a result, a key problem in production management environments is incorporating geologic uncertainty while maintaining operational flexibility.

Combining numerical reservoir models with geological measurements (obtained from either seismic simulations or sensors embedded in reservoirs that dynamically monitor changes in fluid and rock properties) can aid in the design and implementation of optimal production strategies. In that case, the optimization process involves several steps:

(1) simulate production via reservoir modeling;
(2) detect and track changes in reservoir properties by acquiring seismic data (through field measurements or seismic data simulations);
(3) revise the reservoir model by imaging and inversion of output from seismic data simulations.

Figure 2 illustrates the overall optimization process. Oil reservoir simulations are executed using environment values (e.g. permeability of rocks, initial values of oil and gas pressures) and placements of production and injection wells. The datasets generated by the simulations can be analyzed to forecast production. The datasets can also be processed to generate input for seismic data simulations to track changes in rock properties over time. Analysis of seismic datasets (e.g. using seismic imaging algorithms) can reveal how the geophysical characteristics of the reservoir change over short and long periods of time. The data can also be converted into input to the oil reservoir simulators for additional reservoir simulations, with potentially different well placements.

As stated earlier, one challenging problem in the overall process is incorporating geological uncertainty. An approach to address this issue is to simulate alternative production strategies (number, type, timing and location of wells) applied to multiple realizations of multiple geostatistical models. In a typical study, a scientist runs an ensemble of simulations to study the effects of varying oil reservoir properties (e.g. permeability, oil/water ratio, etc.) over a long period of time. This approach is highly data-driven. Choosing the next set of simulations to be performed requires analysis of data from earlier simulations.

Another major problem is the enormity of data volumes to be handled. Large-scale, complex models (several hundreds of thousands of unknowns) often involve multiphase, multicomponent flow, and require the use of distributed and parallel machines. With the help of large PC clusters and high-performance parallel computers, even for relatively coarse descriptions of reservoirs, performing series of simulations can lead to very large volumes of output data. Similarly, seismic simulations can generate large amounts of data. For example, downhole geophysical sensors in the field and ocean bottom seismic measurements are episodic to track fluid changes. However, per episode, these involve a large number (e.g. hundreds to thousands) of detectors and large numbers of controlled sources

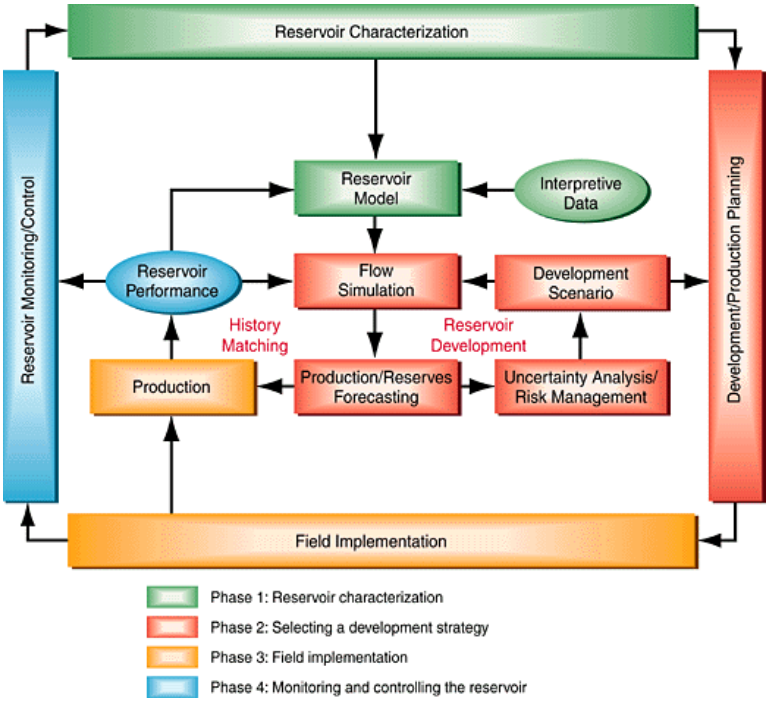


Figure 1. A production management environment.

of energy to excite the medium generating waves that probe the reservoir. These data are collected at various time intervals. Thus, seismic simulators that model typical three-dimensional seismic field data can generate simulation output that is terabytes in size. *As a result, traditional simulation approaches are overwhelmed by the vast volumes of data that need to be queried and analyzed.*

In this work, we develop a software system that is designed to support: (1) accurate and efficient discretization methods for reservoir and geophysical simulations; (2) efficient storage and processing of simulation output on distributed, disk-based active storage systems[‡] to rapidly handle *ad-hoc* queries and data product generation; and (3) techniques for assimilating, analyzing, and interpreting very large and diverse data. We describe the components of the software system and describe the implementation

[‡]PC clusters built from low-cost, commodity items are increasingly becoming widely used. With high-capacity, commodity disks, they create *active storage clusters* that enhance a scientist's ability to store large-scale scientific data. The low cost/performance ratio of such systems makes them attractive platforms for interactive data analysis applications. We refer to them as active storage because not only do they provide a platform for data storage and retrieval, but also some application-specific processing can be carried out on the system.

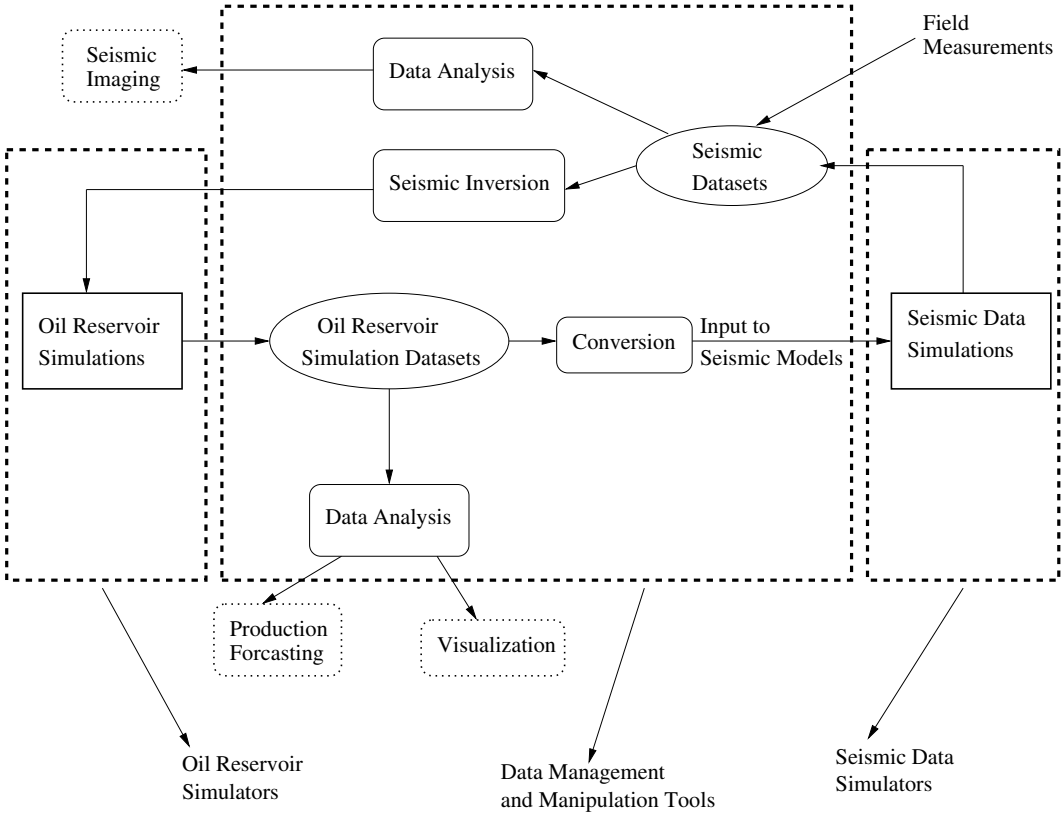


Figure 2. Well management and production optimization loop using numerical models.

of several data analysis scenarios. We present performance results for the data analysis scenarios and for the system components.

In Section 2, we describe the overall system architecture and the individual components of the system in greater detail. Section 3 presents a number of data analysis scenarios and describes how seismic and oil reservoir simulations are coupled to each other. An experimental evaluation of the system components is described in Section 4. Section 5 presents an overview of the related work.

2. AN INTEGRATED SYSTEM FOR OIL RESERVOIR MANAGEMENT

We propose an integrated simulation system (Figure 3) to support large-scale studies for oil production optimization on distributed collections of PC compute and active storage clusters. This system consists of three main subsystems.

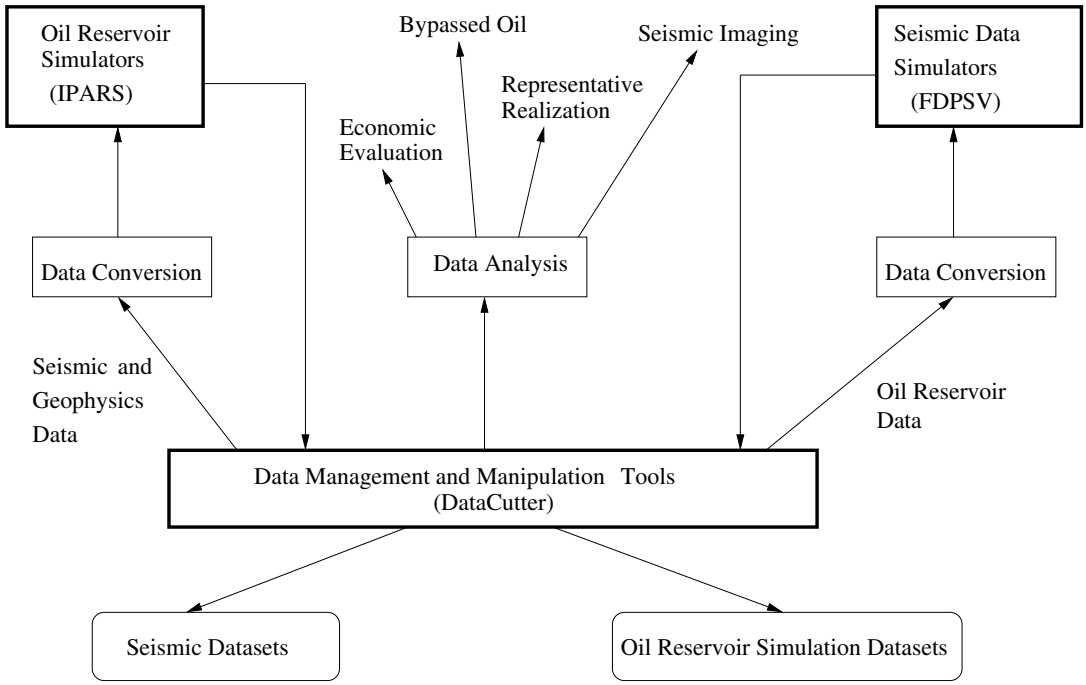


Figure 3. Overall system architecture. The system consists of three main components: oil reservoir simulators, seismic data simulators, and data management and manipulation tools. The specific software components used in this paper are indicated in parentheses in each box. These components are described in detail in Section 2. The data analysis applications (i.e. economic evaluation, bypassed oil, representative realization, and seismic imaging) implemented in this paper using the data management and manipulation tools are also shown in the figure. These applications are described in Section 3.

(1) A simulation framework that supports multiple physical models and algorithms for the solution of multiphase flow and transport problems in porous media (*oil reservoir simulators*).
(2) A simulator for modeling controlled source seismic measurements and the response of the reservoir to those measurements (*seismic data simulators*).
(3) A component-based framework for storage and processing of simulation datasets in a distributed environment (*data management and manipulation tools*).

An architecture of the integrated simulation system is shown in Figure 3. The simulation tools are used to simulate alternative production strategies for a large number of geostatistical realizations and to model subsurface material properties and changes in properties. Output from these simulators is stored on distributed collections of disk-based storage systems for interactive data analysis. In this system, the coupling of seismic simulators and oil reservoir simulators is done through data management and manipulation tools. Output from seismic simulations and oil reservoir simulations is stored and managed by the data management tools. Seismic data simulation tools query and process the datasets

generated by oil reservoir simulations to generate input data and parameter values. Similarly, oil reservoir simulations access the subsets of seismic data output to revise the reservoir characteristics for new simulations.

In addition to the coupling of simulation tools, various data analysis operations can be implemented that query and manipulate datasets of simulation output so as to forecast production amount, assess the economic value of the reservoir under study, and understand the changes in reservoir characteristics through seismic imaging and visualization (Figure 2). These operations can be executed on the storage systems where the datasets are stored or on other machines dispersed across a network. A scientist can formulate queries to carry out different analysis scenarios, such as economic ranking of the alternatives, exploration of the physical basis for differences in behavior between realizations, in particular to identify regions of bypassed oil and representative realizations, which could be used as a basis for further optimization. In the following sections, we describe the individual components of this system in greater detail.

2.1. Oil reservoir simulator: IPARS

The reservoir simulation part of the described work was performed with the use of the Integrated Parallel Accurate Reservoir Simulator (IPARS). IPARS has been developed at the Center for Subsurface Modeling (CSM) [1–4] and is suitable for simulation of various non-trivial models of multiphase, multicomponent flow and transport in porous media. IPARS is portable across several serial and parallel platforms including Linux (clusters), SGI, IBM SP, T3E, Intel IA-32, IBM Power4 and MS Windows. A unique feature of the framework is the multiblock and multiphysics capability, which allows for the computational domain to be decomposed into several subdomains, each with a separate Grid system and with an individual physical and numerical model associated with it [4–10]. Furthermore, a modular and flexible structure of the framework allows for its easy integration with other software; for example, IPARS has been coupled with DISCOVER [11,12] and with NetSolve [13] for interactive steering [14].

The physical models in IPARS range from single-phase through two-phase oil–water and air–water to three-phase black-oil and compositional models. In addition, general reactive transport and compositional multicomponent models are available. Finally, poroelastic models are available [15]. Solvers used by IPARS employ state-of-the-art techniques for nonlinear and linear problems including multigrid and other preconditioners [16].

In this project we use the fully implicit black-oil model implemented under IPARS. This model is a three phase (water, oil and gas) model describing the flow in a petroleum reservoir with three components. Here it is assumed that no mass transfer occurs between the water phase and the other two phases and that the water phase can be identified with the water component. In the hydrocarbon (oil–gas) system, only two components, light and heavy hydrocarbons, are considered. Furthermore, the following standard assumptions have been made in developing the mathematical model: the reservoir is isothermal and its boundaries are impermeable, permeability tensor is diagonal and aligned with the coordinate system, Darcy's law is valid and viscosity of fluids is constant within the assumed range of reservoir pressures and the reservoir rock formation is slightly compressible.

We note that this last assumption can be lifted, as was done, for example, in a loose-in-time coupling of flow and geomechanics processes realized as a connection between the black-oil flow model under IPARS and the elasticity–plasticity geomechanics model under JAS3D [17]. However, such a coupling

is very complex and currently has only limited portability, therefore this coupled model is not used in this work. In the future, a fully integrated coupled poroelastic model [18] will be used. We remark that the model has been shown [19,20] to give accurate results by comparison with (i) available analytical solutions and on a benchmark problem [21], and with (ii) another numerical simulator: a recognized industrial reservoir simulation tool Eclipse [22].

Input to the model

Input to the numerical black-oil model consists of all the data that characterizes the reservoir and its fluids, in particular, the porosity ϕ and permeability K, as well as relative permeability, capillary pressures and compressibility and pressure/volume/temperature (PVT) data. In addition, exploration parameters are specified that define the position and operating conditions of injection and production wells (rates q_M, bottom hole pressure etc.). Finally, the time of simulation and the spatial *ijk* Grid as well as the time-stepping δt_n are specified.

Output

The results of numerical black-oil model are of two distinct types. The first type of data is collected in so-called *well files*, which collect the resulting well rates $q_M^n = q_M(t_n)$, for every time step t_n and every component $M(water, oil, gas)$. Well files are output by one of the processors only, are of low volume and are additionally buffered in order to decrease the time spent on I/O. The second type is the data that can be used for post-processing analysis and visualization purposes and it encompasses pointwise values of field variables. These are defined in the numerical model as piecewise-constant over each cell *ijk*. Such values are output every couple of time steps of simulation, depending on the needs of a user, and the resulting dataset can quickly reach large volume. Some other variables, for example, velocities, are defined at the edges/faces between cells.

2.2. Seismic data simulator: FDPSV

The Finite-Difference Simulation of P and SV Waves (FDPSV) is a simulator for the elastic waves that are used to model the reservoir's response to controlled source seismic measurements. Predicting the elastic response requires the elastic reservoir model and the model for the surrounding geology. Since we expect the fluid properties of the reservoir to change during production, if we can match the seismic predictions to the seismic data recorded, we can infer the changes in the reservoir model.

FDPSV is a fourth order in space and second order in time staggered Grid explicit finite difference solution to the elastic wave equation. It can have sources of seismic energy and receivers anywhere within the computation Grid so it is possible to acquire numerical data for any acquisition geometry. For three-dimensional data simulations, nine tractions and three displacements are computed for each time step.

In our implementation, we map the flow parameters to elastic parameters in the same Grid and thus we make no use of any upscaling. For seismic simulation, data are required at the Grid location specified to model a given seismic acquisition scheme. One approach to parallelization of FDPSV is to distribute different parts of the three-dimensional model to different nodes and make use of domain decomposition. However, we adopt a much simpler strategy in that we simply run multiple FDPSV

simulations on different nodes for different source locations. Each controlled source being modeled is a new run of the algorithm. Many controlled source activations can be run in parallel since in most modern PC clusters each node has adequate memory for a realistic three-dimensional simulation.

2.3. Data management and manipulation tools: DataCutter

IPARS and FDPSV provide the simulation components of the integrated system. As we noted earlier, the operation of the system is highly data driven: In a typical study, multiple oil reservoir simulations are executed using IPARS and FDPSV. Output from these simulations are analyzed for production estimation; analysis of simulation output can also drive new reservoir and seismic simulations. All of these steps involve the storage, management, and processing of large volumes of data. Since these complex, long-running simulations require significant computing power, it is likely that the simulations are executed on multiple, distributed computing platforms. As a result, the datasets are generated and stored in a distributed, heterogeneous environment. A unique aspect of our integrated system is its ability to manage and manipulate very large volumes of oil reservoir and seismic datasets on heterogeneous, distributed collections of storage and compute systems. This ability requires a runtime system that supports efficient indexing and retrieval of large data volumes and execution of data processing operations in a distributed setting. In this work, we employ a component-based framework, called DataCutter [23–25], to address these requirements.

In DataCutter, the data-processing structure of an application is represented as a set of application filters. A *filter* is a user-defined object that performs application-specific processing on data. A filter object has three functions[§], *init*, *process* and *finalize*, that should be implemented by the application developer. A *stream* is an abstraction used for all filter communication, and specifies how filters are logically connected. A stream also denotes a supply of data to and from the storage media, or a flow of data between two separate filters, or between a filter and a client. Bi-directional data exchange can be achieved by creating two pipe streams in opposite directions between two filters. All transfers to and from streams are through a provided buffer abstraction (Figure 4). A buffer represents a contiguous memory region containing useful data. The init function is where any required resources such as memory can be pre-allocated for a filter. The process function of the filter is called by the runtime system to read from any input streams, work on data buffers received, and write to any output streams. The finalize function is called after all processing is finished, to allow the release of allocated resources such as scratch space.

A set of filters that collectively carry out application-specific data processing are referred to as a *filter group*. The runtime system provides a multi-threaded, distributed execution environment. Multiple instances of a filter group can be instantiated and executed concurrently; work can be assigned to any group. Within each filter group instance, multiple copies of individual filters can also be created. Filters co-located on the same machine are executed as separate threads by the runtime system. Data exchange between two co-located filters is carried out by simple pointer copy operations, thereby minimizing the communication overhead. Communication between two filters on different hosts is done through TCP/IP sockets.

[§]The DataCutter framework supplies a base class with three virtual functions (*init*, *process*, *finalize*). An application-specific filter is derived from this base class, and application operations are implemented in the three functions.

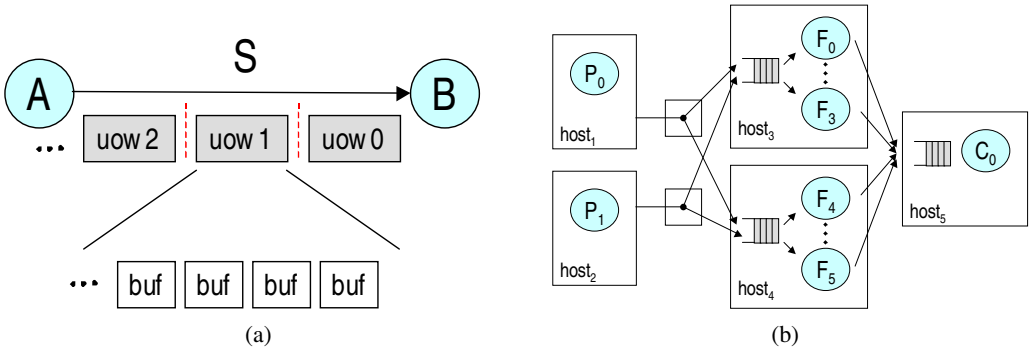


Figure 4. DataCutter stream abstraction and support for copies: (a) data buffers on a stream; (b) P, F, C filter group instantiated using transparent filter copies.

Decomposition of application processing into filters and filter copies effectively provide a combination of task- and data-parallelism (Figure 4(b)). Filter groups and filters can be placed on a distributed collection of machines to minimize computation and communication overheads. Filters that exchange large volumes of data can be placed on the same machine, while compute intensive filters can be executed on more powerful machines or less loaded hosts.

3. PUTTING IT TOGETHER: A CASE STUDY

As is illustrated in Figures 2 and 3, the process of optimizing oil production involves coupling of oil reservoir models and seismic data models and analysis of output from ensembles of simulations in order to predict production rates and geophysical changes. Both the coupling of numerical models and analysis of data are made possible through the data management and manipulation infrastructure.

Section 2 presented the overall system architecture and the individual components of the system. We now present the implementation of a case study. In this case study, the IPARS and FDPSV components of the integrated system are used to simulate oil production using the models described in Sections 2.1 and 2.2. The third component of the system, DataCutter, is a framework that can be employed to implement various data analysis operations on both IPARS and FDPSV datasets as well as data transformation operations to couple IPARS with FDPSV. In this section, we describe the implementation using DataCutter of several data analysis operations and of coupling between IPARS and FDPSV. We start with the implementation of data transformation operations to create FDPSV input from IPARS output. Following that section are the sections on the implementations of data processing applications to analyze datasets generated by IPARS and FDPSV. In Section 4, we carry out a performance evaluation of these implementations using large datasets.

3.1. Data conversion from IPARS to FDPSV

IPARS generates the reservoir parameters during production, and these parameters, in particular saturation and porosity, must be converted into the elastic wave model parameters of compressional wave velocity, shear velocity and density. Empirical formulation based on the Biot–Gassman equations are used to map the reservoir simulator output to the seismic model input.

Different steps of mapping flow parameters to elastic parameters are as follows. If the rock pores have water, oil and gas with varying saturation, effective fluid density is given by the weighted average of the densities ρ_m of the component fluid phases weighed by the fluid saturations S_m

$$\rho_{\text{fluid}} = S_{\text{wat}}\rho_{\text{wat}} + S_{\text{oil}}\rho_{\text{oil}} + S_{\text{gas}}\rho_{\text{gas}}$$
$$S_{\text{wat}} + S_{\text{oil}} + S_{\text{gas}} = 1.0$$

The effective density of the reservoir rock is given by the weighted average of the densities of the solid matrix ρ_{rock} and the density of the fluid filling up the pore space

$$\rho_{\text{eff}} = \phi\rho_{\text{fluid}} + (1 - \phi)\rho_{\text{rock}}$$

Similar averaging is done to compute effective bulk modulus and shear modulus. Therefore, the compressional and shear wave velocities that are functions of elastic moduli and density, can be easily computed. At each time step of IPARS run, we have pore pressure and saturation, and thus we can employ the Biot–Gassman equation to predict elastic properties at different time steps during the production of a hydrocarbon reservoir. These parameters then serve as model parameters to run FDPSV, which computes seismic response.

DataCutter services are used to convert IPARS reservoir simulation output to the elastic properties needed as inputs to the seismic simulator FDPSV. IPARS output is stored in simulation timestep order, and may be partitioned across multiple storage devices, so it is necessary to recombine the distributed data and extract the variables of interest before performing the variable conversion. The converted reservoir data are then rescaled and inserted into a pre-existing seismic model based on a set of user-defined parameters.

The DataCutter implementation consists of a server process located on a machine that can access the metadata that describes the IPARS simulation runs and the locations of their output, and a set of DataCutter filters. Five filters were developed to handle the different portions of the conversion and insertion process.

- The *Provider* filter reads the file containing the original seismic model and forwards it to the filter that ultimately performs the insertion of the converted reservoir data. A client provides the host location of the input seismic model, and a single copy of the Provider filter is mapped to that host.
- The *Reader* filter is designed to access and subset IPARS output based on a given timestep and variable. Instances of the Reader are mapped to each host on which IPARS output is located.
- Since data sent by multiple Reader filters for a timestep may arrive in an arbitrary order, a *Serializer* filter acts as a buffer for combining received data into the correct order. A single copy of this filter is mapped to any available host.
- A *Flattener* filter combines the distributed data into a single dataset and formats it properly for the conversion code.

- The final filter, the *Biot–Gassman* filter, performs the actual conversion and handles the insertion of the resulting reservoir data into the original seismic model. It computes the required compressional wave velocity, shear wave velocity, and the density, from the reservoir variables P_{oil} (oil pressure), S_{oil} (oil saturation), and S_{gas} (gas saturation). This filter is mapped to the host specified as the desired output location.

A single conversion request is initiated by a client query, which selects an available dataset and specifies the locations of the input and output seismic model. After the server process has determined the appropriate mappings for all of the filters, based on the IPARS input and other user-provided information, each request is handled in five stages. In the first stage the filters are initialized with the locations of the input and output files, both for IPARS and FDPSV, and the input seismic model is transferred from the Provider filter to the Biot–Gassman filter and saved to disk on that host. In the second stage, parameters for the IPARS run, such as the problem and Grid size, the number of processors, and a list of the stored timesteps, are read by the server process and distributed to each of the Reader filters so that they can calculate the portion of the problem for which they are responsible and allocate memory accordingly. The Reader filters also search for and use per-processor IPARS initialization files describing the partitioning of the reservoir data across processors to create a partial index, or *data descriptor*, for the data locally available[¶]. During the third phase, the partial descriptors are sent by each Reader to the Flattener, so that a global descriptor can be generated that maps the data indices to their processor locations. The fourth phase begins when the client selects a particular timestep and provides the parameters needed for insertion of the reservoir data into the seismic model. At that time, data for each of the three IPARS variables (P_{oil}, S_{oil}, S_{gas}) for the given timestep is read by each Reader filter, sent to the Serializer filter, which organizes the buffers into processor order, then forwarded to the Flattener for final merging and formatting. In the fifth and final phase, the Biot–Gassman filter performs the conversion and insertion of the reservoir data into the seismic model.

3.2. Analysis of IPARS output

We have implemented several data exploration scenarios. These scenarios involve user-defined queries for economic evaluation as well as technical evaluation, such as determination of representative realizations and identification of areas of bypassed oil.

3.2.1. Economic evaluation

In the optimization of oil field production strategies, the objective function to be maximized is the resulting economic value of a given production strategy. The value can be measured in a variety of ways. In our model we compute both the net present value (NPV) and return on investment (ROI). In the computation of the NPV for a given realization, a query integrates over time the revenue from produced oil and gas, and the expenses from water injection and production, accounting for the time value of the resources produced. This calculation is performed for a subset of the realizations chosen by the user. The user specifies the costs and prices that parameterize the objective function, e.g. oil

[¶]The partitioning of the IPARS data depends on how the IPARS simulator was run in parallel, and how it wrote its output data.

price, cost of handling produced water, drilling costs. As all of the well production and injection data for each realization resides in a single file on a single disk, the data access pattern for this application is relatively simple, and most of the computation time is spent parsing the output file. The well data is also a relatively small part of the output data at each time step, so this is not a compute- and data-intensive computation. Presently, the operations for the economic evaluation are implemented as a single DataCutter filter, which also performs data retrieval from the storage system.

3.2.2. Bypassed oil

Depending on the distribution of reservoir permeability and the production strategy employed, it is possible for oil to remain unextracted from certain regions in the reservoir. To optimize the production strategy, it is useful to know the location and size of these regions of bypassed oil. To locate these regions, we apply the following algorithm. In this algorithm, the user selects a subset of datasets (D), a subset of time steps (T), minimum oil saturation value ($S_{\text{oil,tol}}$), maximum oil velocity ($V_{\text{oil,tol}}$), and minimum number of connected Grid cells (N_{c}) for a bypassed oil pocket. The goal is to find all the datasets in D that have some bypassed oil pockets with at least N_{c} Grid cells. A cell (C) is a potential bypassed oil cell if $S_{\text{oil,c}} > S_{\text{oil,tol}}$ and $V_{\text{oil,c}} < V_{\text{oil,tol}}$.

(1) Find all potential bypassed oil cells in a dataset at time step $T_i \in T$.
(2) Run a connected components analysis. Discard pockets with fewer cells than N_{c}.
(3) Mark cells that are in a region of bypassed oil.
(4) Perform an AND operation on all cells over all time steps in T.
(5) Perform the operations in step (2) on the result, and report back to client.

We used the DataCutter framework to implement a set of filters that carry out the various operations employed by the algorithm. The implementation consists of three filters. RD (Read Data filter) retrieves the data of interest from disk and writes the data to its output stream. A data buffer in the output stream contains oil velocity and oil saturation values, and corresponds to a portion of the Grid at a time step in a data set. CC (Connected Component filter) performs steps (1), (2) and (3) of the bypassed oil algorithm. It carries out operations to find bypassed oil pockets at a time step on data buffer received from RD. These oil pockets are stored in a data buffer, which is a byte array. Each entry of the data buffer denotes a Grid cell and stores if the cell is bypassed oil cell or not. The CC filter writes the data buffer for each time step to the output stream, which connects CC to the MT filter. MT (Merge over Time filter) carries out the steps (4) and (5) of the bypassed oil algorithm. The filter performs an AND operation on the data buffers received from CC, and finds the bypassed oil pockets. The result is sent to the client. Figure 5 shows the implementation of the bypassed oil computations as a filter group and a possible execution of the filter group using the transparent copies abstraction of DataCutter. This scenario accesses the large four-dimensional (three spatial dimensions and time) datasets which are output for each realization.

3.2.3. Representative realization

Running multiple realizations with the same geostatistical model and well configurations can give an idea of the upper and lower bounds of performance for a particular production strategy. It is also of interest to find one realization for a given production scenario that best represents the average or

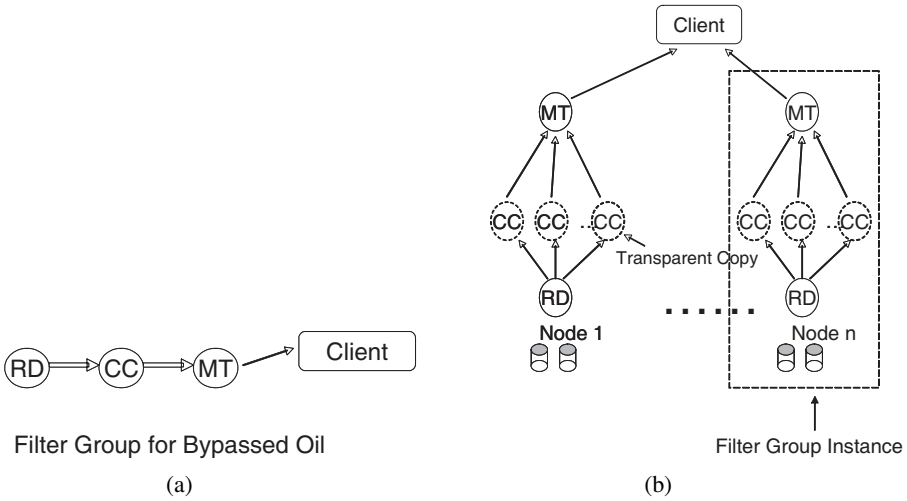


Figure 5. (a) Filter group for bypassed oil computations and (b) execution on a parallel system.

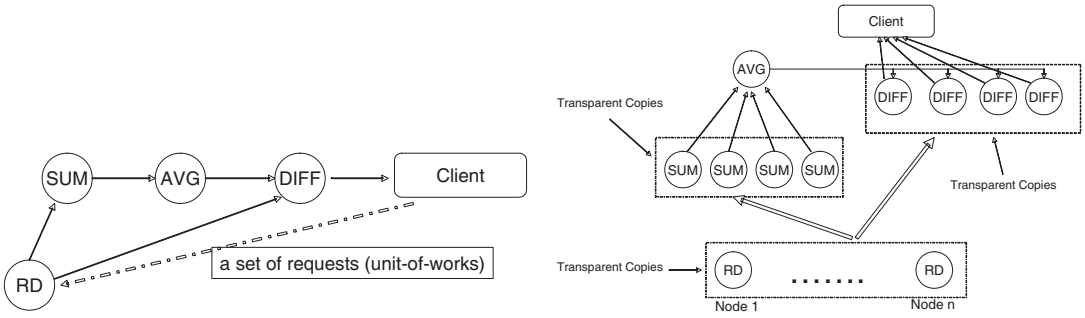
expected behavior. A client query to find a representative realization for a given subset of realizations computes the average of the primary IPARS unknowns (oil concentration C_{OIL}, water pressure P_{wat}, gas concentration C_{GAS}) and then finds the realization in the subset, for which the values at each Grid point is closest to the average values:

$$R_i = \sum_{\text{all Grid points}} \frac{|C_{\mathrm{OIL}} - C_{\mathrm{OIL\_avg}}|}{C_{\mathrm{OIL\_avg}}} + \frac{|P_{\mathrm{wat}} - P_{\mathrm{wat,\_avg}}|}{P_{\mathrm{wat,\_avg}}} + \frac{|C_{\mathrm{GAS}} - C_{\mathrm{GAS,\_avg}}|}{C_{\mathrm{GAS,\_avg}}} \tag{1}$$

$$\text{Representative Realization} = \min(R_i) \quad \text{for } i \in \{\text{subset of realizations}\} \tag{2}$$

The DataCutter implementation consists of four filters. RD retrieves the data of interest from disk. The read filter sends data from each dataset to the SUM (Sum filter) and DIFF (Difference filter) filters. A data buffer from the read filter is a portion of the Grid at one time step. SUM computes the sum for C_{OIL}, P_{wat} and C_{GAS} variables at each Grid point across the datasets selected by the user. AVG (average filter) calculates the average for C_{OIL}, P_{wat} and C_{GAS} values. DIFF finds the sum of the differences between the Grid values and the average values for each dataset. It sends the difference to the client, which keeps track of differences for each time step, carries out average over all time steps for each dataset.

Figure 6 shows the implementation of the representative realization computations as a filter group and a possible execution of the filter group using the transparent copies abstraction of DataCutter on the storage system where the simulation output is stored.



(a) (b)

Figure 6. (a) Filter group for representative realization computations and (b) execution on a parallel system.

3.3. Analysis of FDPSV output: seismic imaging

Seismic data are recorded as sound traces in the field and under simulation in a standard exchange format defined by the Society of Exploration Geophysics. They are used to infer subsurface material properties or changes in properties. Also, the data are used directly to create subsurface images and estimates of material properties. The former is termed model-based inversion, while the latter is called seismic imaging and direct inversion.

Seismic imaging analysis requires that subsets of seismic data be selected based on the type of sensor in a recording array, the actual recording array or arrays, and for each or a suite of source activations. Thus, we must be able to quickly sort through data sets of terabyte scale. Multiple queries will have to be made as we interpret or analyze each dataset. In order to speed up the selection of data to be used in imaging, an index can be built on the dataset. For this purpose, we form a key from the values of recording array, source activations and the type of sensor in a recording array. A B-tree index is built using these key values. Pointers to the corresponding sound traces in the dataset are also stored along with the key values. For a client query, this index is searched to find the pointers to the data elements (sound traces) that satisfy the query. Then, the selected data elements are retrieved from the data files and input to the seismic imaging algorithm.

The seismic imaging algorithm first receives the subsurface velocity model and then using an integral solution to the wave equation computes the travel times that describe the travel path from the source to the receiver using a subsurface scattering or diffraction model. The data along these trajectories in the data space are then summed into the image point to form a numerical stationary phase evaluation for the current velocity model. If the model is close to the actual, events will add in phase and a coherent image will be constructed. Born inversion consists of applying weights to the data to be imaged based on their angle of incidence to give an estimate of the fractional changes in, e.g. the compressional wave velocity. The overall algorithm for seismic imaging can be represented by the following pseudo-code.

(1) Extract a subset of seismic traces from the dataset.
(2) Allocate and initialize three-dimensional velocity and image volume.

(3) **foreach** shot point p **do**
(4) **read** all traces for p.
(5) **add** the contribution of each trace to the image volume (time migration).
(6) **enddo**

For execution in a parallel or distributed environment, we implement two DataCutter filters. An *Aggregation filter* (A) performs the time migration computations. A *Reader filter* (R) retrieves the traces for shot points and sends them to the worker filters. Multiple copies of the Reader filter can be created to read data in parallel if the dataset is partitioned and distributed across multiple storage nodes. Similarly, multiple copies of the Aggregation filter can be instantiated to parallelize the processing of traces. The three-dimensional image volume is employed as an accumulator that maintains the partial contributions as the traces are retrieved and processed. We should note that the operation (time migration) performed at step (5) of the algorithm is commutative and associative. That is, the final output value is the same irrespective of the order the traces are retrieved and their contribution is added to the output. This aspect of the algorithm allows for different parallelization strategies [26,27]. In this paper we consider two parallelization strategies, *replicated accumulator* and *partitioned accumulator* [26].

In the partitioned accumulator strategy, the three-dimensional image volume is partitioned across the copies of the Aggregation filter. The partitioned accumulator strategy makes good use of the aggregate system memory by partitioning the image volume, but it results in communication overhead. Since each shot trace contributes to every element in the image volume, a Reader filter has to send a trace to all the copies of the Aggregation filter.

In the replicated accumulator strategy, the three-dimensional image volume is replicated in each copy of the Aggregation filter. A Reader filter can send a shot trace to any one of the copies of the Aggregation filter. The replicated accumulator strategy can be implemented using the transparent copies abstraction of DataCutter and can take advantage of the demand-driven mechanism employed in DataCutter. The three-dimensional image volume in each Aggregation filter contains partial results at the end of processing. These results should be merged to compute the final output. Thus, a *Merge filter* (M) that combines the partial results from copies of the Aggregation filter is part of the filter group in the replicated accumulator strategy. This strategy reduces the volume of communication that would be incurred because of input shot traces. However, it does not make efficient use of distributed memory in the system as it replicates the data structures for the three-dimensional image volume data in all Aggregation filters, and introduces communication and computation overhead because of the Merge filter. Figure 7 illustrates the two strategies.

4. PERFORMANCE EVALUATION

In this section we report on the performance evaluation of the implementations of data analysis and data transformation operations, described in Section 3. The performance of the simulation tools in the context of computational complexity and parallel scalability are described in [10,16,28,29]. Briefly, realistic reservoir simulation cases are large, do not fit in the memory of a single workstation and have to be simulated on massively parallel platforms. The mechanism underlying the parallel features

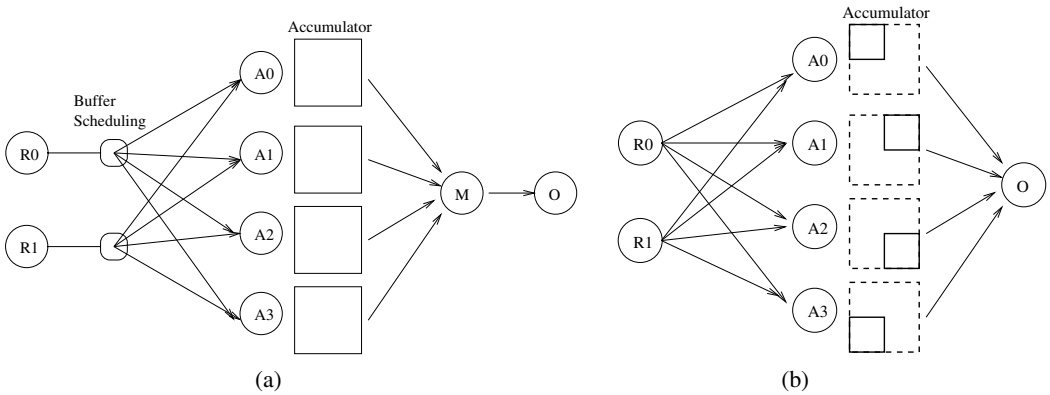


Figure 7. (a) The replicated accumulator strategy. The application is composed of Reader (R), Aggregation (A) and Output (O) filters. The Merge filter (M) is added to ensure correct execution when multiple copies are executed. Buffers from a R filter are dynamically scheduled among A filters using a demand-driven policy. (b) The partitioned accumulator strategy. In this example, the accumulator structure is partitioned into four disjoint regions, and each region is assigned to one of the copies of filter A. Since the accumulator is partitioned, a merger filter is not needed. An output filter O simply receives results from each copy of filter A and stitches them together to form the final output.

of the IPARS framework are routines based on MPI [30], which realize the updates of ghost cells as well as the global-reduce type operations. Load-balancing is Grid-based whereby the cells are distributed evenly between processors, or model-based [10]. If multiple realizations are simulated, then the computational time required to complete the overall set of simulations scales linearly with the number of simulations.

4.1. Case study datasets

In order to evaluate the performance of the data analysis applications, we constructed three case study datasets. The first two were oriented towards the data analysis of multiple realizations of reservoir simulation data, each of which was of small or of medium size. The third was designed to provide a meaningful test case for connecting IPARS and FDPSV. Here we briefly summarize the characteristics of these cases.

4.1.1. Industry benchmark-based multiple realizations: case `spe9`

The input data for this dataset is based on the industry benchmark SPE9 problem [21], which requires the solution of a black-oil (three phase) flow problem on a Grid with $9000 = 24 \times 25 \times 15$ cells, with 26 wells, and with highly heterogeneous permeability field. The base case is very well known and is used frequently as a test for nonlinear and linear solvers used in reservoir simulators. In our project we used the base case and we generated additional permeability fields using a fast Fourier transform

(FFT)-based geostatistics program provided by Jim Jennings from Bureau of Economic Geology at University of Texas at Austin [31]. Additionally, we varied the well patterns. The total of 207 realizations were taken from among 18 geostatistical models and four well configurations/production scenarios. The geostatistical models are used to randomly generate permeability fields that are characterized by statistical parameters such as covariance and correlation length.

At each time step, the value of 17 separate variables is output for each node in the Grid. A total of 10 000 time steps were taken and the total output stored for each realization is about 6.9 GB. The total dataset size is roughly 1.5 TB and was generated and stored on a storage cluster, referred to here as UMD, of 50 Linux nodes (PIII-650, 768 MB, Switched Fast Ethernet) with a total disk storage of 9 TB at University of Maryland.

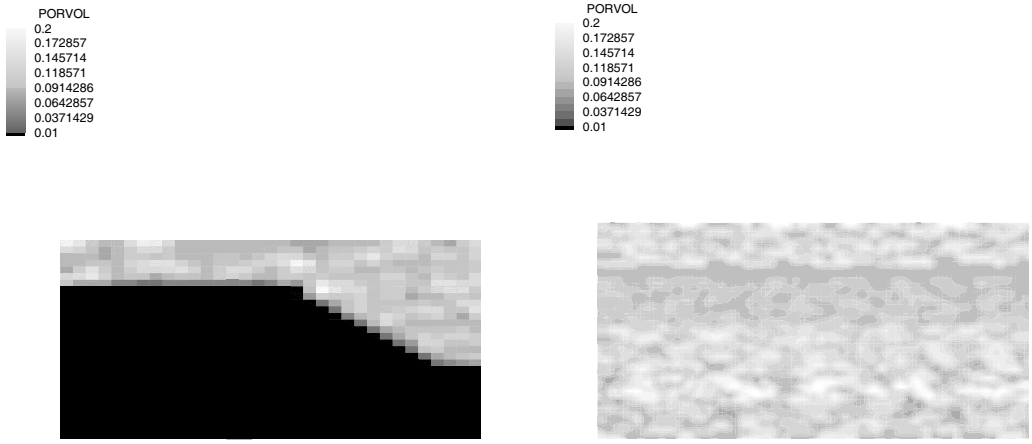
4.1.2. GSLIB-based multiple realizations: case `bh`

This case was an extension of the `spe9` dataset. The base case was $65\,536 = 64 \times 64 \times 16$ cells with one or five wells, with some limited refinement patterns applied uniformly to the whole Grid. The permeability fields were created using constrained sequential Gaussian simulation (`sgsim`) program from the GSLIB family [32]. The use of conditional stochastic simulations, that is the use of *prior* knowledge of reservoir data in the vicinity of wells, decreased the uncertainty of the simulation results as well as improved the number of successful simulations with respect to the `spe9` cases. Each realization simulated 17 variables over 2000 time steps and generated an output of 10 GB. A total of 500 realizations were executed, resulting in a dataset of 5 TB. This dataset was generated and stored at three locations: The High-Performance Storage System (HPSS) at San Diego Supercomputer Center, UMD at the University of Maryland and a 24-node storage system, referred to here as OSUMED, with 7.2 TB of disk space at the Ohio State University.

4.1.3. Case study for coupled IPARS–FDPSV simulations: case `bitr`

The purpose of this case was to construct a meaningful example for testing the connection between IPARS and FDPSV simulations. In other words, we considered only one realization of permeability field, a large enough case and production parameters capable of generating significant variation in seismic properties. The reservoir model is $3000' \times 6000 \times 300'$ with a uniform Grid of $50 \times 100 \times 30$ cells. The porosity ϕ and permeability K fields for this case were constructed to create a (stretched) Z-shaped almost impermeable fault, which separates two types of sand: sand type 1 and sand type 2 (see Figure 8). The flow in the reservoir is driven by six water injection wells; the fluids (oil, water and gas) are produced from five production wells. Injection wells are located under the upper arm of the 'Z' (depth 80–170′), and production wells are above the lower arm of the 'Z'. The simulation was over 20 000 days.

Due to previous experience [17], we determined that the presence of free gas phase is essential in order to generate significant gradients of pressure, porosity, as well as of saturations, which become data for the Biot–Gassmann equations. In addition, our synthetic case had significant geological features so we could determine how well the seismic simulation would be able to detect them. The reservoir simulation results are shown in Figure 9. The figure presents the contours of the



(a) (b)

Figure 8. Synthetic reservoir model for `bitr` case: (a) the side view of the porosity field; (b) the top view of the porosity field.

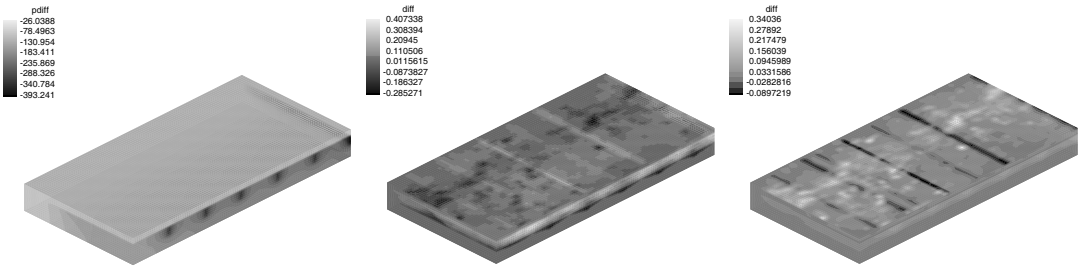


Figure 9. Results of reservoir model. The contours of difference of final and initial values of pressure and of oil and gas saturations are shown.

differences of P_{wat}, S_{oil}, S_{gas} between the beginning and the end of production. Seismic simulation results for the entire seismic domain, which is much larger than the reservoir model, are shown in Figure 10. These show the change in compressional velocity and correlate very well with the structure of porosity of the data set.

4.2. IPARS to FDPSV conversion

Figure 11 shows the processing times for conversion of IPARS output data to FDPSV input. The case study used in this experiment was `bitr` described in Section 4.1.3. The values were obtained by

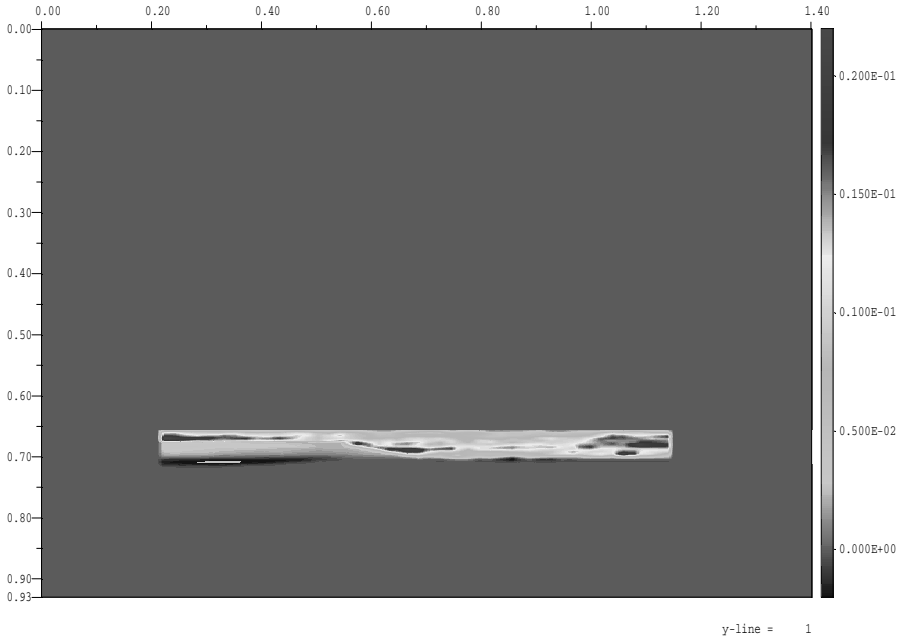


Figure 10. Change in compressional wave velocity after the completion of production of the hydrocarbon reservoir. A two-dimensional slice is shown here. The Biot–Gassman equation was used to convert the IPARS output parameters to elastic parameters.

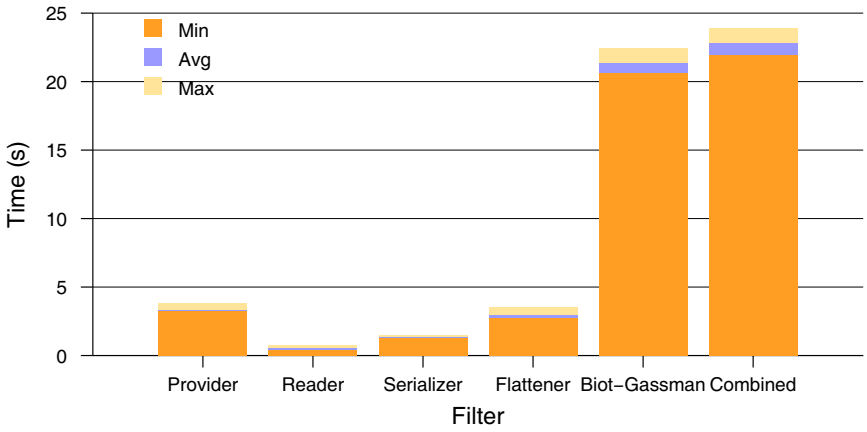
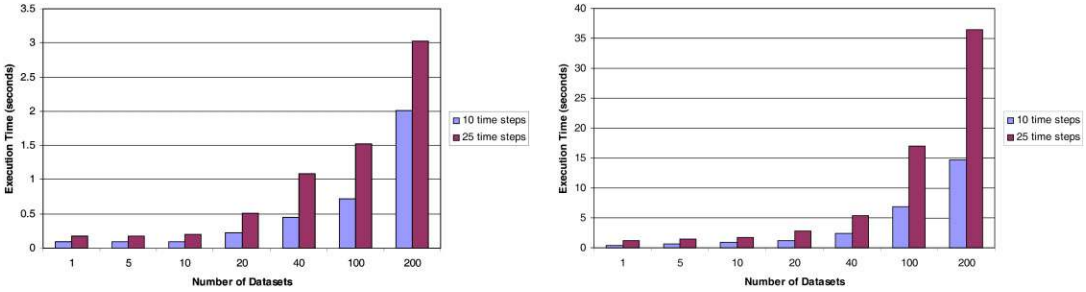


Figure 11. IPARS to FDPSV data conversion.



(a) (b)

Figure 12. Performance numbers using the dataset generated in case `spe9`: (a) performance results for bypassed oil computation; (b) performance results for representative realization computation.

running five conversion queries on a 57.7 GB IPARS dataset distributed evenly across 20 650 MHz Pentium III nodes (2.9 GB per node). A single reader filter was assigned to each of these nodes. The Provider and Serializer filters ran on a dual processor 400 MHz Pentium II data where the 32 MB input seismic model resided. The Flattener and Biot–Gassman filters ran on an eight-processor 550 MHz Pentium III.

The graph shows the minimum, maximum and average processing time per filter, and the last column shows the values for overall query execution times. The combined value, on average, is approximately 22% less than the sum of the individual filter timings, because of overlap in processing between the various stages of the DataCutter pipeline. The relative times of the first four filters correlate directly with the amount of data sent to the next stage in the pipeline by the filter. On average, the Provider sends the 32 MB seismic model to the Biot–Gassman filter, the Readers sends 408 KB to the Serializer, the Serializer sends 4.3 MB to the Flattener and the Flattener forwards 4.3 MB to the Biot–Gassman filter. The Biot–Gassman filter is by far the most expensive stage of the pipeline, because of the large amount of computation required to perform the variable conversion.

4.3. Analysis of output from IPARS

We have carried out experiments using the bypassed oil and representative realization scenarios (see Section 3.2). In the first set of experiments, analysis queries were executed on the datasets generated in case `spe9` (see Section 4.1.1) and stored at University of Maryland. The second set of experiments involved queries to the datasets generated in case `bh` (see Section 4.1.2) and stored at University of Maryland and the Ohio State University.

In Figure 12, we present results for the first set of experiments. The queries were evaluated on 19 nodes of the 9.5 TB storage cluster at University of Maryland. In the experiments, we varied the number of data sets accessed by a query. For this purpose, we submitted a total of 28 queries (varying the number of datasets from 1 to 200); 14 queries for the bypassed oil analysis and 14 queries for the representative realization analysis. Half of the queries in each set requests data over 10 time steps (time

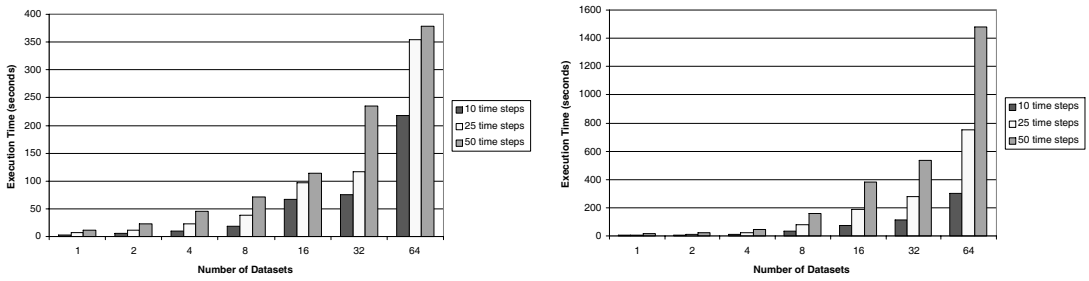


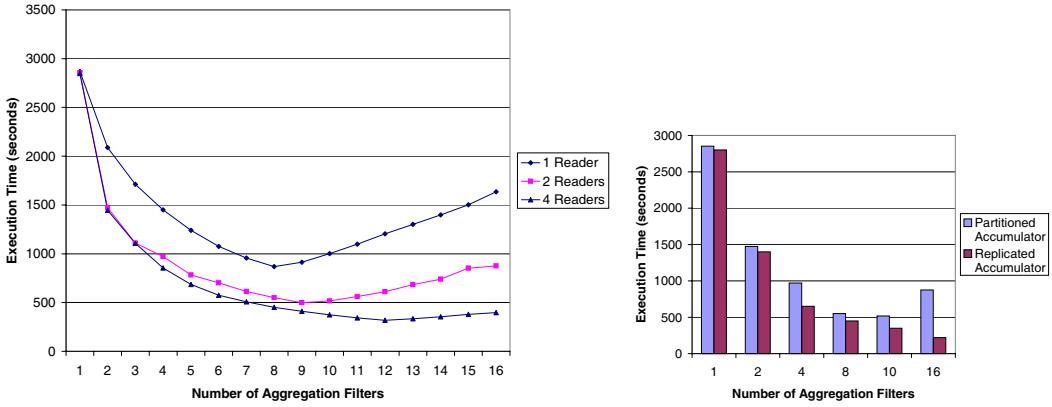
Figure 13. Performance numbers using the dataset generated in case `bh`: (a) performance results for bypassed oil computation; (b) performance results for representative realization computation.

steps 0 through 9999, with increments of 1000 time steps), while the other half retrieves 25 time steps (time steps 0 through 9999 with increments of 400 time steps).

As is seen from the figure, up to 40 datasets the query execution time remains below one second for the bypassed oil scenario. As the number of datasets is increased the query execution time increases, as expected. For a query that accesses 200 datasets over 25 time steps, the execution time is about three seconds. Thus, we are able to achieve interactive rates even for queries that access a large number of datasets from the collection. The experimental results show that queries for representative realization scenario take longer, as the operations involved are more expensive. As is seen from the figure, the query execution time remains below five seconds for queries that access up to 40 datasets over 25 time steps. Our preliminary results show that the query execution does not scale well after 40 datasets for the representative realization scenario. This is because of the fact that in the experiments the number of transparent copies for the SUM and DIFF filters are fixed at four.

In the second set of experiments, queries requested data from 68 realizations, where output from 20 of the realizations were stored on eight nodes of the cluster at University of Maryland and output from 48 of the realizations were stored on 16 nodes of the cluster at the Ohio State University. The nodes of each cluster are connected over a switched 100 Mbits s^{-1} Fast Ethernet. The connection between the two clusters is over the Internet. We observed about 1.1 Mbits s^{-1} bandwidth between the two clusters. We should note that the size of the Grid used in this set of experiments is about seven times as large as than that of the Grid in the first set of experiments, a single time step consists of a Grid of 65 536 elements. Another difference of this dataset is each simulation had been run in parallel on eight nodes. Each node stored the local portion of the output, effectively partitioning the output of each realization across those nodes.

Figure 13 displays average execution time of the queries. Since we do not have exclusive control of the wide-area network, one would expect to see some variations in the execution time of the queries due to the variations in the network performance. To alleviate this problem to some extent, we repeated each experiment three times. The performance numbers in the figures are the average of the three runs. As is seen from the figure, the execution time of bypassed oil analysis queries does not increase linearly



(a) (b)

Figure 14. (a) Execution time of the partitioned accumulator strategy, when the number of reader and aggregation filters is varied. (b) Execution times of the replicated accumulator and partitioned accumulator strategies when the number of aggregation filters is varied. The number of readers is two.

with the number of time steps, as expected. There are a number of reasons for this behavior. One of them is that since this is not a compute intensive operation and setup time for the query is significant, increasing the number of time steps that should be processed does not increase the execution time linearly. Second, the bypassed oil scenario is very well suited for pipelined execution of the data; as a result, it does not have a synchronization point like the average filter in the implementation of representation realization scenario. This second set of experimental results also show that queries for the representative realization scenario take longer, as the operations involved are more expensive.

4.4. Analysis of output from FDPSV: seismic imaging

We have evaluated the relative performance of the two strategies described in Section 3.3 for imaging of seismic simulation output. The experiments were carried out using a 1 GB dataset generated by FDPSV. The three-dimensional image volume was 1.3 MB in size. Each node of the PC cluster used in these experiments had one 933 MHz Pentium III processor and 512 MB main memory. The nodes of the cluster were connected via Switched Fast Ethernet.

Figure 14(a) shows the execution time of the partitioned accumulator strategy when the number of reader and aggregation filters is varied. In this experiment, the input dataset is partitioned across one, two, and four nodes. The number of reader filters is equal to the number of nodes that store the input dataset. As is seen from the figure, the execution time decreases, as the number of reader filters is increased (i.e. the dataset is distributed across more nodes). We attribute this performance improvement to the fact that as the dataset is distributed among more nodes, the aggregate I/O

bandwidth increases. The graph also shows that for a given distribution of the dataset, increasing the number of aggregation filters decreases the execution time. However, after a minimum point, the execution time begins to increase. This is because in the partitioned accumulator strategy, a shot trace is sent to all the aggregation filters. As more aggregation filters are added to the configuration, the volume of communication increases, eventually offsetting the gain obtained by parallelization of aggregation operations.

Figure 14(b) compares the performance of the partitioned and replicated accumulator strategies. In this experiment, the input dataset is distributed across two nodes. As is seen in the figure, the replicated accumulator strategy performs better than the partitioned accumulator strategy. Although the three-dimensional image volume is replicated, its size is much smaller compared with the input dataset size. Moreover, a reader filter sends a shot trace to only one of the copies of the aggregation filter. As a result, the volume of communication in the replicated accumulator is smaller than that in the partitioned accumulator strategy.

5. RELATED WORK

Requirements associated with the need to access geographically dispersed data repositories have been a key motivation behind several large data Grid projects. The Biomedical Informatics Research Network (BIRN) [33] and Shared Pathology Informatics Network (SPIN) [34] are examples of projects that target shared access to medical data in a wide-area environment. GriPhyN project [35] targets storage, cataloging and retrieval of large, measured datasets from large-scale physical experiments. The goal is to deliver data products generated from these datasets to physicists across a wide-area network. The objective of Earth Systems Grid (ESG) project [36] is to provide Grid technologies for storage, publishing and movement of large-scale data from climate-modeling applications. The TeraGrid [37] effort aims to develop an integrated systems software suite capable of managing a scalable set of Grid-based data and compute resources. The EUROGRID project [38] intends to develop tools for easy and seamless access to high-performance computing (HPC) resources. MEDIGRID [39] is another project recently initiated in Europe to investigate application of Grid technologies for manipulating large medical image databases. The Asia Pacific BioGRID [40] is an initiative that aims to provide an Asia-Pacific-wide computational resource package that will benefit all biomedical scientists in the Asia Pacific.

Driven by such large-scale applications, a large body of research has been focused on developing middleware frameworks, protocols and programming and runtime environments. These efforts have led to the development of middleware tools and infrastructures such as Globus [41], Condor-G [42], Storage Resource Broker [43], Network Weather Service [44], DataCutter [24,45,46], Legion [47], Cactus [48], and Common Component Architecture [49], and many others [50–55]. There were also been some recent efforts to develop Grid and Web services [56–59] implementations of database technologies [60–63].

Our project is directed towards the general problem of modeling and characterization of the Earth's subsurface. Thus, it has immediate application to other areas, including environmental remediation, aquifer management and storage of hazardous wastes. To the best of our knowledge, no integrated systems exist to address computational and data-handling challenges in large-scale oil reservoir studies.

6. CONCLUSIONS

We have presented a new paradigm for applying reservoir simulation to the challenges of reservoir management. The selected challenge is to enable the analysis and evaluation of large numbers of realizations, both of geological models and of production strategies. We have proposed a software system to address this challenge. In this system, the IPARS and FDPSV frameworks provide the numerical solutions to the forward-flow problems, while the DataCutter middleware provides the means for subsetting and filtering the multidimensional output. The volume of data resulting from such studies can be extremely large. Such datasets would be unmanageable for most evaluation tools, especially for complex queries such as identifying representative realizations or locating regions of bypassed oil, or analysis of variation of seismic parameters such as the input to the Biot–Gassman equation.

The integrated simulation and data analysis system proposed in this paper enables the creation, interrogation and visualization of such datasets while maintaining the familiarity and speed of interaction of the traditional simulation workflow. Thus, many more realizations of higher-resolution geologic models and more production strategies can be studied in greater detail within a given time, increasing the utility of the study for decision making.

ACKNOWLEDGEMENTS

We would like to thank Steven Bryant for his involvement with the `spe9` case and for his invaluable input for the data-analysis scenarios. We are also very grateful to Yifeng Cui and Nancy Wilkins-Diehr of the San Diego Supercomputer Center (SDSC) for their help in generating the datasets at SDSC and to SDSC for providing access to IBM SP Blue Horizon computer and HPSS for running the simulations and storing the datasets.

This research was supported in part by the National Science Foundation under Grants #ACI-9619020 (UC Subcontract #10152408), #ANI-0330612, #EIA-0121177, #SBR-9873326, #EIA-0121523, #ACI-0203846, #ACI-0130437, #ACI-9982087, NPACI #10181410, Lawrence Livermore National Laboratory under Grant #B517095 (UC Subcontract #10184497), NIH NIBIB BISTI #P20EB000591, Ohio Board of Regents BRTTC #BRTT02-0003, and DOE #DE-FG03-99ER2537.

REFERENCES

1. IPARS: Integrated Parallel Accurate Reservoir Simulator. http://www.ticam.utexas.edu/CSM/ACTI/ipars.html [2 August 2001].
2. Parashar M, Wheeler JA, Pope G, Wang K, Wang P. A new generation EOS compositional reservoir simulator. Part II: Framework and multiprocessing. *Proceedings of the 14th SPE Symposium on Reservoir Simulation*. Dallas, TX, June 1997. Society of Petroleum Engineers: Houston, TX, 1997; 31–38.
3. Wang P, Yotov I, Wheeler MF, Arbogast T, Dawson CN, Parashar M, Sepehrnoori K. A new generation EOS compositional reservoir simulator. Part I: Formulation and discretization. *Proceedings of the 14th SPE Symposium on Reservoir Simulation*, Dallas, TX, June 1997. Society of Petroleum Engineers: Houston, TX, 1997; 55–64.
4. Wheeler MF, Wheeler JA, Peszyńska M. A distributed computing portal for coupling multi-physics and multiple domains in porous media. *Computational Methods in Water Resources*, Bentley LR, Sykes JF, Brebbia CA, Gray WG, Pinder GF (eds.). A. A. Balkema, 2000; 167–174.
5. Peszyńska M, Lu Q, Wheeler MF. Coupling different numerical algorithms for two phase fluid flow. *MAFELAP Proceedings of Mathematics of Finite Elements and Applications*, Whiteman JR (ed.). Brunel University: Uxbridge, U.K., 1999; 205–214.
6. Peszyńska M, Lu Q, Wheeler MF. Multiphysics coupling of codes. *Computational Methods in Water Resources*, Bentley LR, Sykes JF, Brebbia CA, Gray WG, Pinder GF (eds.). A. A. Balkema: Amsterdam, 2000; 175–182.

7. Wheeler MF. Advanced techniques and algorithms for reservoir simulation, II: The multiblock approach in the integrated parallel accurate reservoir simulator (IPARS). *Resource Recovery, Confinement, and Remediation of Environmental Hazards* (*IMA Volumes in Mathematics and its Applications*, vol. 131), Chadam J, Cunningham A, Ewing RE, Ortoleva P, Wheeler MF (eds.). Springer: Berlin, 2002.
8. Peszyńska M. Advanced techniques and algorithms for reservoir simulation III. Multiphysics coupling for two phase flow in degenerate conditions. *Resource Recovery, Confinement, and Remediation of Environmental Hazards* (*IMA Volumes in Mathematics and its Applications*, vol. 131), Chadam J, Cunningham A, Ewing RE, Ortoleva P, Wheeler MF (eds.). Springer: Berlin, 2002; 21–40.
9. Lu Q, Peszyńska M, Wheeler MF. A parallel multi-block black-oil model in multi-model implementation. *Society of Petroleum Engineers Journal* 2002; **7**(3):278–287 (SPE 79535).
10. Lu Q, Parashar M, Peszynska M, Wheeler MF. Parallel implementation of multi-physics multi-block for multi-phase flow in subsurface. In preparation.
11. Mann V, Matossian V, Muralidhar R, Parashar M. Discover: An environment for web-based interaction and steering of high-performance scientific applications. *Concurrency and Computation: Practice and Experience* April 2001.
12. Muralidhar R, Kaur S, Parashar M. *Architecture for Web-based Interaction and Steering of Adaptive Parallel/Distributed Applications* (*Lecture Notes in Computer Science*). Springer: Berlin, 2000; 1332–1339.
13. Casanova H, Dongara J. Netsolve: A network server for solving computational science problems. *Proceedings of the 1996 ACM/IEEE Supercomputing Conference*. IEEE Computer Society Press: Los Alamitos, CA, 1996.
14. Wheeler MF, Lee W, Dawson CN, Arnold DC, Kurc T, Parashar M, Saltz J, Sussman A. Parallel computing in environment and energy. *Sourcebook of Parallel Computing*, Dongarra J, Foster I, Fox G, Gropp W, Kennedy K, Torczon L, White A (eds.). Morgan Kaufmann: San Mateo, CA, 2002; 145–165.
15. Gai X, Dean R, Wheeler M, Liu R. Coupled geomechanical and reservoir modeling on parallel computers. *SPE Reservoir Simulation Symposium*, Houston, TX, 3–5 February 2003. Society of Petroleum Engineers: Houston, TX, 2003 (SPE 79700).
16. Lacroix S, Vassilevski Y, Wheeler JA, Wheeler MF. An iterative solution of linear systems in the implicit parallel accurate reservoir simulator (IPARS). In preparation.
17. Minkoff S, Stone CM, Bryant S, Peszyńska M, Wheeler MF. A loose coupling algorithm for fluid flow and geomechanical deformation modeling. *Journal of Petroleum Science and Engineering* 2003; **38**(1–2):37–56.
18. Gai X. A coupled geomechanics and reservoir flow model on parallel computers. *PhD Thesis*, University of Texas at Austin, Austin, TX, 2003.
19. Lu Q. A parallel multi-block/multi-physics approach for multi-phase flow in porous media. *PhD Thesis*, University of Texas at Austin, Austin, TX, 2000.
20. Lu Q, Peszynska M, Gai X. Implicit black-oil model in IPARS framework. *TICAM Report 01-33*, University of Texas at Austin, Austin, TX, 2001.
21. Killough JE. Ninth SPE comparative solution projection: a reexamination of black-oil simulation. *Proceedings of the 13th Symposium on Reservoir Simulation*. San Antonio, TX, February 1995. Society of Petroleum Engineers: Houston, TX, 1995; 135–147 (SPE 29110).
22. Schlumberger Technology Corporation. *Eclipse-100 Reference Manual*, 1998.
23. Beynon M, Kurc T, Sussman A, Saltz J. Design of a framework for data-intensive wide-area applications. *Proceedings of the 9th Heterogeneous Computing Workshop (HCW2000)*. IEEE Computer Society Press: Los Alamtos, CA, 2000; 116–130.
24. Beynon MD, Ferreira R, Kurc T, Sussman A, Saltz J. DataCutter: Middleware for filtering very large scientific datasets on archival storage systems. *Proceedings of the 8th Goddard Conference on Mass Storage Systems and Technologies/17th IEEE Symposium on Mass Storage Systems NASA/CP 2000-209888*. National Aeronautics and Space Administration: Moffett Field, CA, 2000; 119–133.
25. Beynon MD, Kurc T, Sussman A, Saltz J. Optimizing execution of component-based applications using group instances. *IEEE International Symposium on Cluster Computing and the Grid (CCGrid2001)*, Brisbane, Australia, May 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001.
26. Chang C, Kurc T, Sussman A, Catalyurek U, Saltz J. Optimizing retrieval and processing of multi-dimensional scientific datasets. *Proceedings of the Third Merged IPPS/SPDP (14th International Parallel Processing Symposium & 11th Symposium on Parallel and Distributed Processing)*. IEEE Computer Society Press: Los Alamitos, CA, 2000.
27. Chang C, Kurc T, Sussman A, Catalyurek U, Saltz J. A hypergraph-based workload partitioning strategy for parallel data aggregation. *Proceedings of the Eleventh SIAM Conference on Parallel Processing for Scientific Computing*. SIAM: Philadelphia, PA, 2001.
28. Wheeler MF, Peszyńska M, Gai X, El-Domeiri O. Modeling subsurface flow on PC cluster. *High Performance Computing*, Tentner A (ed.). The Society for Modeling and Simulation International (SCS), 2000; 318–323.
29. Peszyńska M, Wheeler MF, Yotov I. Mortar upscaling for multiphase flow in porous media. *Computational Geosciences* 2002; **6**:73–100.

30. Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J. *MPI: The Complete Reference*. MIT Press: Cambridge, MA, 1996.
31. Jennings JWJ, Ruppel SC, Ward WB. Geostatistical analysis of permeability data and modeling of fluid-flow effects in carbonate outcrops. *Society of Petroleum Engineers Reservoir Evaluation and Engineering* 2000; **3**(4):292–303.
32. Deutsch CV, Journel AG. *GSLIB: Geostatistical Software Library and User's Guide* (*Applied Geostatistics Series*). Oxford University Press: Oxford, 1997.
33. Biomedical Informatics Research Network (BIRN). http://www.nbirn.net [December 2004].
34. Shared Pathology Informatics Network (SPIN). http://spin.nci.nih.gov [December 2004].
35. Grid Physics Network (GriPhyN). http://www.griphyn.org [January 2005].
36. Earth Systems Grid (ESG). http://www.earthsystemgrid.org [January 2005].
37. TeraGrid. http://www.teragrid.org [December 2004].
38. EUROGRID. http://www.eurogrid.org/ [May 2004].
39. MEDIGRID. http://creatis-www.insa-lyon.fr/MEDIGRID/home.html [2003].
40. Asia Pacific BioGrid. http://www.apgrid.org [May 2004].
41. The Globus Project. http://www.globus.org [December 2004].
42. Frey J, Tannenbaum T, Foster I, Livny M, Tuecke S. Condor-G: A computation management agent for multi-institutional Grids. *Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC10)*. IEEE Press: Piscataway, NJ, 2001.
43. SRB: The Storage Resource Broker. http://www.npaci.edu/DICE/SRB/index.html [August 2004].
44. Wolski R, Spring N, Haves J. The network weather service: A distributed resource performance forecasting service for metacomputing. *Journal of Future Generation Computing Systems* 1999; **15**(5-6):757–768.
45. Beynon MD, Kurc T, Catalyurek U, Sussman A, Saltz J. Efficient manipulation of large datasets on heterogeneous storage systems. *Proceedings of the 11th Heterogeneous Computing Workshop (HCW2002)*. IEEE Computer Society Press: Los Alamitos, CA, April 2002.
46. Beynon MD, Kurc T, Catalyurek U, Chang C, Sussman A, Saltz J. Distributed processing of very large datasets with DataCutter. *Parallel Computing* 2001; **27**(11):1457–1478.
47. Grimshaw AS, Wulf WA, the Legion Team. The Legion vision of a worldwide virtual computer. *Communications of the ACM* 1997; **40**(1):39–45.
48. Allen G, Dramlitsch T, Foster I, Karonis N, Ripeanu M, Seidel E, Toonen B. Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus. *Proceedings of the 2001 ACM/IEEE SC01 Conference*. ACM Press: New York, 2001.
49. Common Component Architecture Forum. http://www.cca-forum.org [December 2004].
50. Oldfield R, Kotz D. Armada: A parallel file system for computational. *Proceedings of CCGrid2001: IEEE International Symposium on Cluster Computing and the Grid*, Brisbane, Australia, May 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001.
51. Allcock B, Bester J, Bresnahan J, Chervenak AL, Foster I, Kesselman C, Meder S, Nefedova V, Quesnel D, Tuecke S. Secure, efficient data transport and replica management for high-performance data-intensive computing. *Proceedings of IEEE Mass Storage Conference*, April 2001. IEEE Computer Society Press: Los Alamitos, CA, 2001.
52. Vazhkudai S, Tuecke S, Foster I. Replica selection in the Globus data Grid. *CCGRID'01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*. IEEE Computer Society Press: Los Alamitos, CA, 2001.
53. Sato M, Nakada H, Sekiguchi S, Matsuoka S, Nagashima U, Takagi H. Ninf: a network based information library for a global world-wide computing infrastructure. *Proceedings of HPCN'97* (*Lecture Notes in Computer Science*, vol. 1225). Springer: Berlin, 1997; 491–502.
54. Casanova H, Dongara J. Applying Netsolve's network-enabled server. *IEEE Computational Science and Engineering* 1998; **5**(3):57–67.
55. Thain D, Basney J, Son S, Livny M. Kangaroo approach to data movement on the Grid. *Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC10)*. IEEE Computer Society Press: Los Alamitos, CA, 2001.
56. Foster I, Kesselman C, Nick JM, Tuecke S. The physiology of the Grid: An Open Grid Services Architecture for distributed systems integration, 2002. http://www.globus.org/research/papers/ogsa.pdf [December 2004].
57. Foster I, Kesselman C, Nick J, Tuecke S. Grid services for distributed system integration. *IEEE Computer* 2002; **36**(6):37–46.
58. Cerami E. *Web Services Essentials*. O'Reilly: Sebastopol, CA, 2002.
59. Graham S, Simeonov S, Boubez T, Davis D, Daniels G, Nakamura Y, Neyama R. *Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI*. SAMS Publishing: Indianapolis, IN, 2002.
60. Data Access and Integration Services. http://www.cs.man.ac.uk/grid-db/documents.html [October 2003].
61. Raman V, Narang I, Crone C, Haas L, Malaika S, Mukai T, Wolfson D, Baru C. Data access and management services on Grid. http://www.cs.man.ac.uk/grid-db/documents.html [October 2003].

62. Bell WH, Bosio D, Hoschek W, Kunszt P, McCance G, Silander M. Project Spitfite—towards Grid Web service databases. http://www.cs.man.ac.uk/grid-db/documents.html [October 2003].
63. Smith J, Gounaris A, Watson P, Paton NW, Fernandes AA, Sakellariou R. Distributed query processing on the Grid. http://www.cs.man.ac.uk/grid-db/documents.html [October 2003].

