Engineering with Computers
DOI 10.1007/s00366-006-0035-9

ORIGINAL ARTICLE

# Models, methods and middleware for grid-enabled multiphysics oil reservoir management

**H. Klie · W. Bangerth · X. Gai · M. F. Wheeler · P. L. Stoffa · M. Sen · M. Parashar · U. Catalyurek · J. Saltz · T. Kurc**

Received: 19 April 2005 / Accepted: 1 February 2006


**Abstract** Meeting the demands for energy entails a better understanding and characterization of the fundamental processes of reservoirs and of how human made objects affect these systems. The need to perform extensive reservoir studies for either uncertainty assessment or optimal exploitation plans brings up demands of computing power and data management in a more pervasive way. This work focuses on high performance numerical methods, tools and grid-enabled middleware systems for scalable and data-driven computations for multiphysics simulation and decision-making processes in integrated multiphase flow applications. The proposed suite of tools and systems consists of (1) a scalable reservoir simulator, (2) novel stochastic optimization algorithms, (3) decentralized autonomic grid middleware tools, and (4) middleware systems for large-scale data storage, querying, and retrieval. The aforementioned components offer enormous potential for performing data-driven studies and efficient execution of complex, large-scale reservoir models in a collaborative environment.

**Keywords** Reservoir simulation · Multiphysics · Grid computing · Optimization · Data management · Large-scale computing

H. Klie (✉) · X. Gai · M. F. Wheeler
Center for Subsurface Modeling, The University of Texas at Austin, Austin, TX 78712, USA
e-mail: klie@ices.utexas.edu

P. L. Stoffa · M. Sen
Institute for Geophysics, The University of Texas at Austin, Austin, TX 78759-8500, USA

M. Parashar
TASSL, Department of Electrical and Computing Engineering, Rutgers University, Piscataway, NJ, USA

U. Catalyurek · J. Saltz · T. Kurc
Department of Biomedical Informatics, Ohio State University, Columbus, OH 43210, USA

W. Bangerth
Department of Mathematics, Texas A&M University, College Station, TX 77843-3368, USA

## 1 Introduction

Simulations oriented to accurately and efficiently predict the flow of oil and gas in subsurface reservoirs is transcendental in hydrocarbon exploitation. Throughout several years, the constant evolution of computing power has allowed specialists to increase the resolution of models and the inclusion of increasingly more complex processes taking place in the reservoir. The inherent complexity, heterogeneity and dynamism of oil reservoirs, however, require new approaches to developing applications for management and understanding of these systems. Current technologies are pushing the envelope to view the reservoir system as a data-driven framework capable of managing and adapting itself based on their current state, available information and their execution context. Moreover, this data-driven framework should be such that actionable information can be efficiently extracted from large volumes of results generated by complex numerical models and large quantities of data gathered by sensors.

Therefore, a dynamic, data-driven applications system approach offers great potential to address such complex problems as understanding and management

of reservoir systems. The fundamental process in this paradigm is dynamic interactions between numerical models, optimization processes, and data. Dynamic data-driven approaches are increasingly becoming more feasible because of the confluence of several technologies. First, advanced sensor technologies have improved our ability to capture data at higher resolution and faster. Second, grid computing is making possible to realize large-scale, complex numerical models. Grid computing infrastructure aims to dynamically and seamlessly link powerful and remote resources to support the execution of large scale and disparate processes characterizing a particular problem. In order to harness wide-area network of resources into a distributed system, a large body of research has been focused on developing grid middleware frameworks, protocols, and programming and runtime environments. These efforts have led to the development of middleware tools and infrastructures such as Globus [1], Condor-G [2], Storage Resource Broker [3], Network Weather Service [4], DataCutter [5], Legion [6], Cactus [7], and Common Component Architecture (CCA) [8], and many others [9–16]. Initiatives devoted to analyze the potentialities in grid computing for lowering infrastructure costs and impacting the economics of technical computing in the oil industry have been increasingly reported in latest conferences and journals; see e.g., special issue in The Leading Edge [17–21], and our previous work [22–25]. Third, large storage space is becoming more affordable thanks to off-the-shelf inexpensive disk units and storage clusters built from commodity items.

Despite these technological advances, effective implementation of dynamic and data-driven approaches for reservoir modeling and reservoir studies requires several challenging issues be addressed. The massive use of grid computing in diverse energy and environmental applications is still in its embryonic stages. Moreover, the design and implementation of high performance, efficient software systems for managing and analyzing large volumes of data (ranging from a few terabytes to multiple petabytes in size) is still a challenging problem.

This paper addresses efficiency issues by exploiting advanced computational techniques for the autonomic, seamless and distributed processing of intensive numerical computations and management of large amounts of data in reservoir simulation studies. We illustrate how the latest advances in numerical methods and tools, grid-enabled autonomic computing middleware, and large-scale data management systems have driven the conception of new paradigms for the accurate modeling of large fields. To that end, we consider (1) multiphysics applications that imply the coupling of flow, geomechanics, petrophysics and seismics; (2) the determination of optimal well locations; (3) the efficient uncertainty management and, (4) the flow/seismic data management. In all these cases, we rely on the integrated parallel accurate reservoir simulator (IPARS) which is based on a multiblock approach for performing scalable simulation of multiphysics, multiscale and multialgorithm reservoir applications. The overall approach includes the interplay of IPARS with discover/automate (a decentralized and autonomic grid middleware service), geosystems data access and management (GeoDAM; a data subsetting and filtering system), Seine/MACE (multiblock adaptive computational engine), and two very efficient stochastic optimization algorithms such as the SPSA (simultaneous perturbation stochastic approximation) and the VFSA (very fast simulated annealing) to work in a coordinated fashion for the data driven intense challenge for achieving optimal exploitation plans.

This work shows how the conjunction and software engineering of these components offers the possibility of developing large-scale efficient approaches to perform uncertainty analysis and studies leading to opportune reservoir management decisions. We believe that the approaches discussed here can also be applied to other fields such as environmental remediation and biomedical tissue engineering.

## 2 Multiphysics oil reservoir management framework

In this section, we present a framework to support dynamic data-driven reservoir management. This data driven multiphysics simulation framework (DDMSF) comprises accurate, multi-resolution, multiphysics models derivable from diverse data sources, coupled with dynamic data-driven optimization strategies for uncertainty estimation and decision-making. Traditionally, the estimation of model parameters and the optimization of decision parameters have been treated separately in decision-making applications. Moreover, most optimization frameworks have been built under the assumption of perfect knowledge of (noise-free) data, forcing specialists to further tune the data when results do not describe the phenomenon under study. This process is unreliable and inefficient in practice, and does not provide, in most cases, a fully unbiased measurement of uncertainty. The DDMSF aims at generating a functional and closely connected feedback loop between data and simulation, driven by optimization.

The technical approach in the development of the DDMSF is divided in three major components: the dynamic decision system (DDS), the dynamic data-driven assimilation system (DDA) and the autonomic grid middleware (AGM). The orchestration of these components provide the computational feedback between data and the model through optimization (see Fig. 1).

*Dynamic decision system* This module utilizes the current knowledge of a subsurface system as input and initiates the decision-making process. It also includes estimates of the reliability and accuracy of proposed strategies, taking into account both numerical errors as well as uncertainty in subsurface characterization. This involves the formulation of objective functions, forward numerical simulation models and optimization algorithms.

The goals include the optimal scheduling, design and deployment of observations (e.g., wells) to optimize a desired response (e.g., optimum economic revenue, minimum bypassed oil); and (2) fitting numerical model output to field measured values (e.g., history matching, seismic data and/or resistivity data fitting together with well constraints). Finding the optimum is often an ill-posed problem. Furthermore, due to the complexity of running forward models, optimization methods must minimize the number of function evaluations. Method that have been used with particular success are a variant of the simulated annealing algorithm, the VFSA [26, 27] and the SPSA method. [28, 29]

*Dynamic data-driven assimilation and analysis system* Data are acquired from different sources and at different times and scales. From the data management and integration perspective, the simulation and optimization components interact with both data generated by simulations and data collected by field sensors. They are driven by the assimilation and analysis of these datasets. For example, if the information available about the subsurface is insufficient, then all simulations will be unreliable and result in large error margins. Subsurface characterization with geophysical and fluid measurements involves the quantitative assessment of the 3D spatial distribution of material properties such as density, P- and S-wave velocities, electrical resistivity, permeability, porosity, magnetic polarization, pressures, or temperatures, from a finite set of noisy measurements. 4D seismic surveys are of increasing use in industry for reservoir characterization. This is combined with reservoir modeling that leads to seismic simulations and predictions. Only by combining 4D seismics and reser-

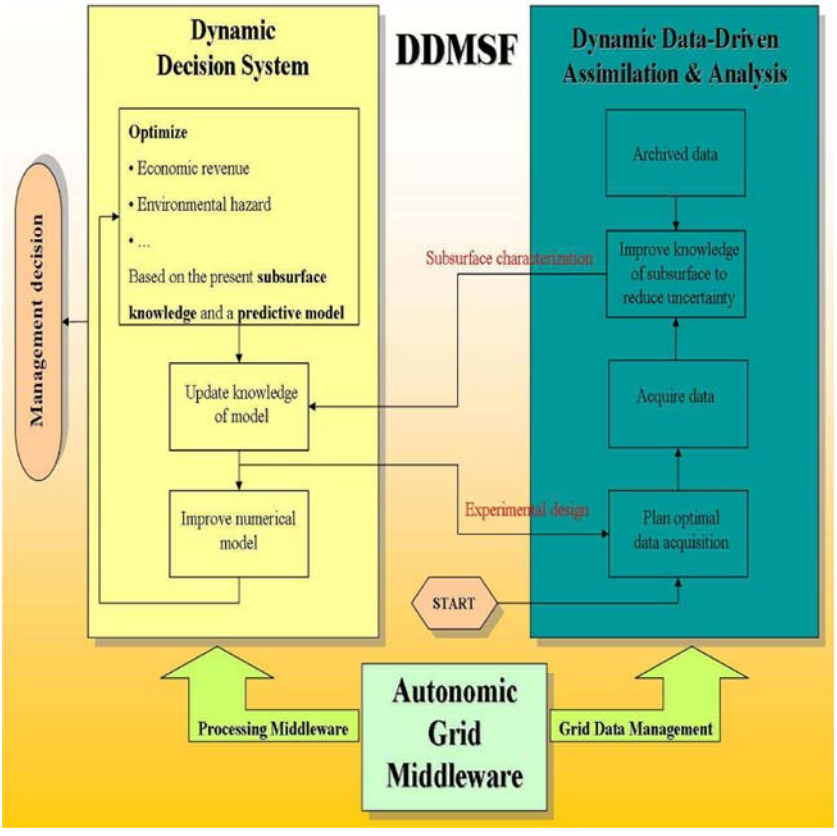


**Fig. 1** Data driven multiphysics simulation framework for subsurface characterization and oil reservoir management

voir simulations can pressure and production data be matched with greater confidence as ambiguities due to fault transmissibility and sand body connectivity are reduced.

The datasets in subsurface characterization and oil reservoir management are large, multi-scale, and they are generated or collected at disparate locations. The dynamic data assimilation and analysis component provides support for data management, data integration, and data processing for the analysis, interpretation, storage and retrieval of these large and heterogeneous data sets.

*Autonomic grid middleware* Distributed computation engines and adaptive runtime management strategies are required to support efficient and scalable implementations of adaptive geophysical and flow reservoir simulations in heterogeneous, widely distributed and highly dynamic grid environments. Control networks with embedded software sensors and actuators are required to enable computational components to be accessed and managed externally, both interactively and using automated policies, to support runtime monitoring, dynamic data injection and control. Self-managing middleware services are necessary to enable seamless interactions where the application components, grid services, resources (systems, CPUs, instruments, storage) and data (archives, sensors) can interact symbiotically and opportunistically as peers. These middleware services must support autonomic behaviors so that interactions and feedback between scales, models, simulations, sensor data and history archives can be orchestrated using high-level policies and rules to navigate the parameter space and optimize subsurface modeling.

In summary, the optimal oil production process involves (1) the use of an integrated multi-physics/multi-block reservoir model (encompassing flow, geomechanics, petrophysics and seismics) and several numerical optimization algorithms (global, local and hybrid approaches) executing on distributed computing systems on the grid; (2) distributed data archives for historical, experimental (e.g., data from field sensors) and simulated data; (3) grid services that provide secure and coordinated access to the resources and information required by the simulations; (4) external services that provide data, such as current oil market prices, relevant to the optimization of oil production or the economic profit; and (5) the actions of scientists, engineers and other experts, in the field, the laboratory, and in management offices. Figure 2 illustrates the interaction of all these components for the optimal reservoir management.

In this process, item 1 is implemented by means of IPARS. The reservoir simulator IPARS is a parallel framework for modeling coupled multiphase flow, reactive transport and geomechanics [30–39]. An attractive feature of IPARS is that it allows for the coupling of different models in different subdomains with possibly non-matching grids [31, 35, 40, 41]. Both forward modeling (comparison of the performance of different reservoir geostatistical parameter scenarios) and inverse modeling (searching for the optimal decision parameters) can greatly benefit from integration and analysis of simulation, historical, and experimental data (item 2). Common analysis scenarios in optimization problems in reservoir simulations involve economic model assessment as well as technical evaluation of changing reservoir properties (e.g., the amount of bypassed oil, the concentration of oil and water). In a grid environment, data analysis programs need to access data subsets on distributed storage systems [42, 43]. This need is addressed by GeoDAM (see Sect. 5). The discover/automate [22, 24, 44] autonomic grid middleware provides the support for items 3, 4, and 5. This middleware couples different components in a peer-to-peer fashion for oil reservoir optimization studies. The peer components involved include sophisticated simulation tools (e.g., IPARS and its coupling with other physical models), a factory component responsible for configuring simulations, executing them on resources on the grid, and managing their execution; optimization service (e.g., VFSA, SPSA and other optimization algorithms); economic modeling services that use simulation outputs and current market parameters (oil prices, costs, etc.) to compute estimated revenues for a particular reservoir configuration. These entities dynamically discover and interact with one another as peers to achieve the overall application objectives.

The main middleware components of the framework are shown in Fig. 3. In this framework, an expert can use portals to interact with the Discover middleware to discover and allocate appropriate resource, and to deploy the factory, optimization service and economic model peers. Optimizer instances can get their initial inputs from data subsets provided by the GeoDAM middleware components. During the optimization process, multiple experts can collaboratively interact with optimizer instances and IPARS. After simulations and optimization processes are completed and the results stored in the environment, additional analysis of the results can be carried out using GeoDAM data subsetting and data processing capabilities.

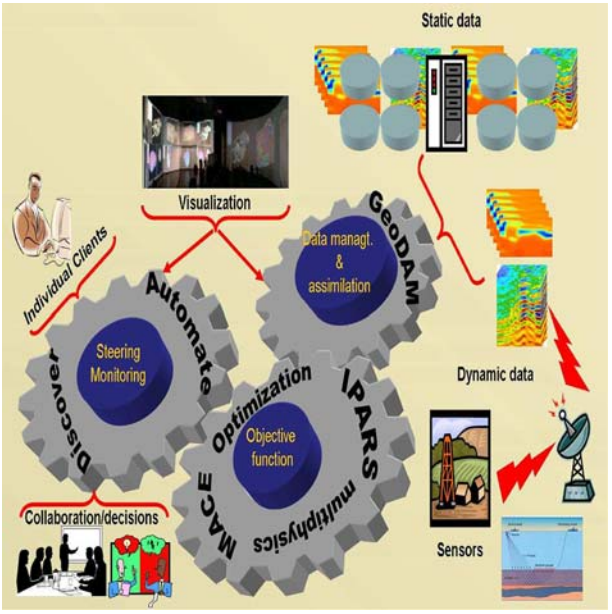


**Fig. 2** Interaction of the several components in DDMSF

## 3 Description of the multiphysics components and optimization

In this section, we briefly describe the numerical models that constitutes the multiphysics model in IPARS: different flow models, petrophysical description, geomechanics and seismics. We also devoted a brief description of the SPSA and VFSA approaches as two novel stochastic optimization algorithms suitable for large-scale implementations. The incorporation of all these models into a single simulation and optimization unit entails the efficient application of the right physics at the right region of the reservoir domain. This also aids at providing a continuous workflow for finding optimal operating scenarios (in terms of profitability, safety or environmental impact) in contrast to the traditional view of integrating a more prolonged assemble of tasks in oil reservoir decision making. Figure 4 illustrates how the integration of different processes into a single multiphysics unit may simplify and speedup the decision loop via optimization.

### 3.1 Reservoir simulation: flow model

The subsurface model consists of a complex interaction of fluid and rock properties that evolves with time. To be able to achieve the desired efficiency and accuracy in the representation of the different phenomena that take place in the subsurface, IPARS offers sophisticated simulation components encapsulating complex mathematical models of the physical interaction in the subsurface, such as geomechanics, chemical reactions, different porous flow processes (single phase, oil–water, air–water, three-phases, compositional), solution

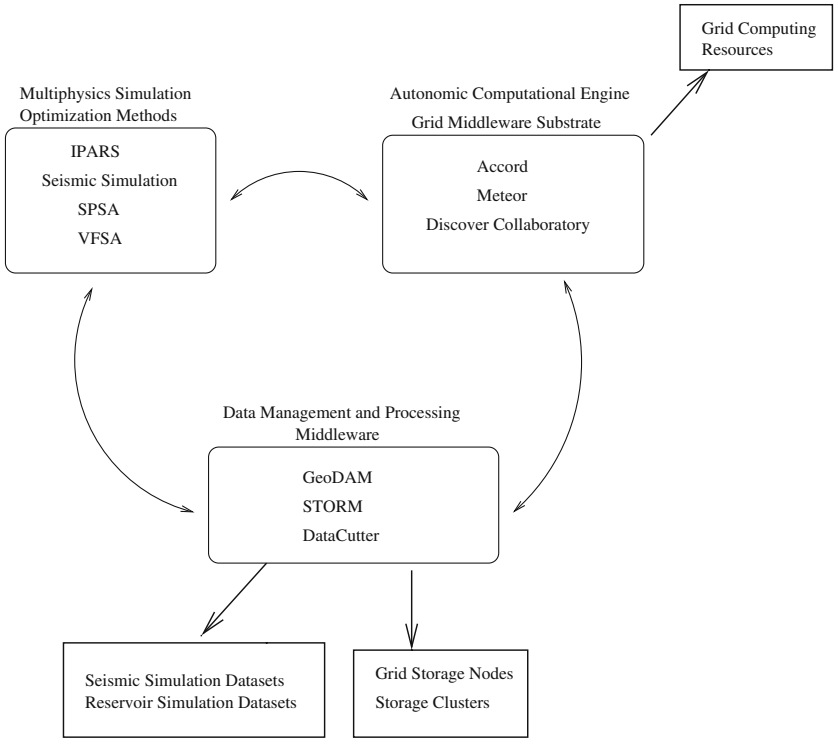


**Fig. 3** The tools, methods, and middleware components of the dynamic data driven multiphysics simulation framework for subsurface characterization and oil reservoir management

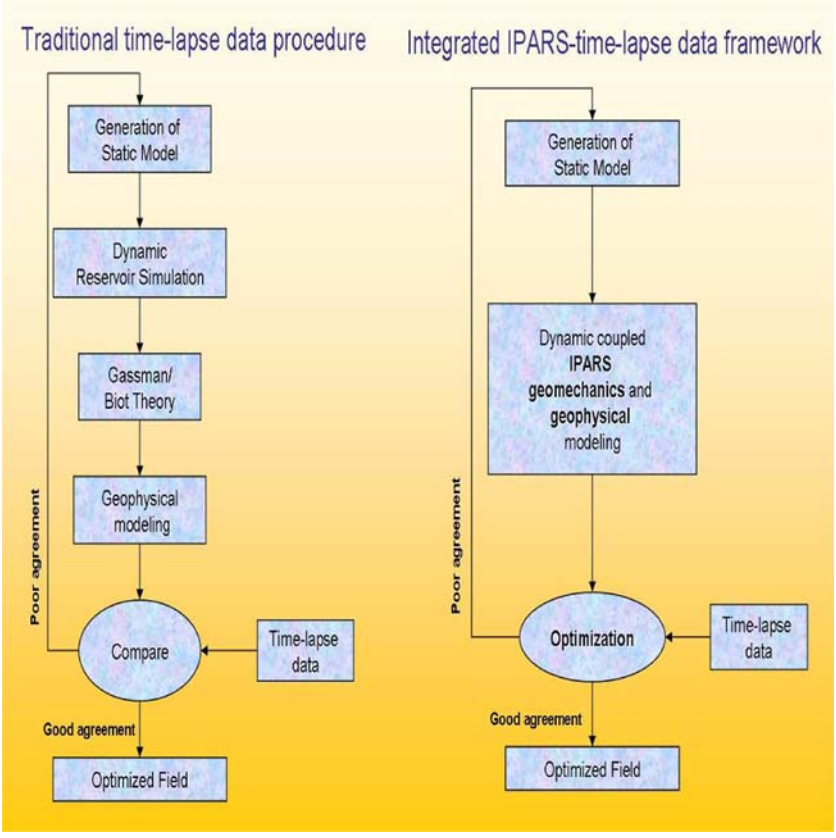


**Fig. 4** Integrating multiple processes for the optimized oil management

algorithms (IMPES, fully implicit) and scales [30–39]. An attractive feature of IPARS is that it allows for the coupling of different models in different subdomains with possibly non-matching grids [31, 35, 40, 41]. It uses state-of the-art solvers and runs on parallel and distributed systems. Solvers for nonlinear and linear problems include Newton–Krylov methods enhanced with multigrid, two-stage and physics-based preconditioners [45]. It can also handle an arbitrary number of wells each with one or more completion intervals.

### 3.2 Multiblock and Seine/MACE

*The multiblock approach* From the conceptual and computational standpoint, different models and flow interactions may take place in the same domain at different spatial and temporal scales. In order to deal with the accurate and efficient solution of these problems, the spatial physical domain is decomposed (i.e., decoupled) in different blocks or subdomains. Domain decomposition algorithms with non-overlapping domains provide an useful approach for spatial coupling/decoupling. A subsurface flow example is the multiblock mortar methodology described in [35, 38, 40, 46–48]. This approach allows for the coupling of different physical processes in a single simulation. Physically driven matching conditions are imposed on block interfaces in a numerically stable and accurate way using *mortar finite element* spaces.

Some of the computational advantages of the multiblock approach are as follows: (1) *multiphysics*, different physical processes/mathematical models in different parts of the domain may be coupled in a single simulation (e.g., coupling single-phase, two-phase, and three-phase flows); (2) *multinumerics*, different numerical techniques may be employed on different subdomains (e.g., coupling mixed finite element and discontinuous Galerkin (DG) methods, explicit, adaptive implicit and fully implicit formulations); (3) *multiscale resolution and adaptivity*, highly refined regions may be coupled with more coarsely discretized regions, dynamic grid adaptivity may be performed locally on each block; and (4) *multidomains*, highly irregular domains may be described as unions of more regular and locally discretized subdomains with the possibility of having interfaces with non-matching grids. The latter allows for the construction of grids that follow large-scale geological features such as faults, heterogeneous layers, and other internal boundaries. This is critical for discretization accuracy. In addition, the appropriate choice of physical models and numerical methods can reduce substantially the

computational cost with no loss of accuracy. The multiblock approach leads to coarse level parallel computations of a domain decomposition type, i.e., it may be implemented efficiently on massively parallel computers with near optimal computational load balance and minimal communication overhead. Figure 5 illustrates the capabilities of the multiblock approach.

When coupling multiple physics and/or multiple domains (which may have their own grid and timestep) through interfaces, one must develop appropriate transmission or matching conditions on the interface. One approach is the use of mortar finite element methods [31, 33, 34, 39–41, 49–51]. The interfaces between blocks are filled with mortars, elements of a finite element space called the mortar space. Mortar finite elements also lend themselves to multiscale resolution, as one can couple highly refined regions where one wants to capture fine scale phenomena, with more coarsely refined regions through the use of a mortar space [35], thus allowing for nonmatching grids between subdomains. A posteriori error estimates for MMFE methods and algorithms for adapting the mortar and subdomain grids have been developed in [52]. It is worth to add, that besides supporting the use of mortar elements, IPARS also comprises discretizations based on DG approximations for the purpose of coupling different physical phenomena and/or different grids [53].

*Seine/MACE shared-space interaction framework and multiblock computational engine* A key challenge presented by the multiblock formulations described above are the dynamic and complex communication and coordination patterns resulting from the multiphysics, multinumerics, multiscale and multidomain couplings. These communication/coordination patterns depend on the state of the subsurface phenomenon being modeled are determined by the specific numerical formulation, domain decomposition and/or subdomain refinement algorithms used, etc., and are known only at runtime. Implementing these communication and coordination patterns using commonly used parallel programming frameworks is non-trivial. Message passing frameworks such as MPI [54], which are the most widely used paradigm, require matching sends and receives to be explicitly defined for each interaction. Programming frameworks based on shared address spaces provide higher-level abstractions that can support dynamic interactions. However, scalable implementation of global shared address spaces remains a challenge.

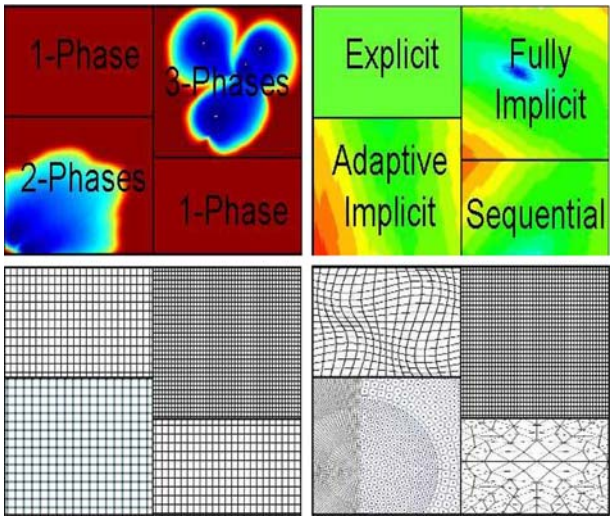


**Fig. 5** An illustration of the multiblock paradigm; from left to right, from top to bottom: multiphysics, multialgorithm, multiscale and multidomain

Associative shared spaces (e.g., tuple spaces) have been shown to provide a very flexible and powerful mechanism for extremely dynamic communication and coordination patterns [55]. In this model, processes interact by sharing tuples in an associative shared tuple space. A tuple is a sequence of fields, each of which has a type and contains a value. The producer of a message formulates the message as a tuple and places it into the tuple space. The consumer(s) can associatively look up relevant tuples using pattern matching on the tuple fields. The tuple space model provides two fundamental advantages: simplicity and flexibility. The communicating nodes need not care about who produced or will consume a tuple. Furthermore, the communicating processes do not have to be temporally or spatially synchronized. This decoupling allows the model to effectively support dynamic communication/coordination. Additionally, the model is accompanied with a general coordination language, such as Linda, which defines a set of extremely simple and clear primitives that present a friendly interface to programmers. However, scalable implementation of tuple spaces remains a challenge. In a pure tuple space environment, all the communication passes through a logically centralized tuple space with a relatively slow associative lookup mechanism [56], which is an inherent bottleneck impeding scalability.

Seine/MACE [57, (L. Zhang and M. Parashar, submitted)] provides a dynamic geometry-based shared space model to support parallel multiblock simulations by building on the tuple space model and extending it to support geometry-based object sharing semantics, space dynamism, and scalable realizations. The Seine model builds on two key observations: (a) formulations of most scientific and engineering applications are

based on geometric multi-dimensional domains (e.g., a grid or a mesh) and (b) interactions in these applications are typically between entities that are geometrically close in this domain (e.g., neighboring cells, nodes or elements). Rather than implementing a general and global associative space, Seine defines geometry-based transient interaction spaces, which are dynamically created at runtime, and each of which is localized to a sub-region of the global geometric domain. Each transient interaction space is defined to cover a closed region of the application domain described by an interval of coordinates in each dimension. The interaction space can then be used to share objects between nodes whose computational sub-domains geometrically intersect with that region. To share an object using the interaction space, nodes do not have to know of, or synchronize with each other at the application layer. Sharing objects in the Seine model is similar to that in a tuple space model. Furthermore, multiple shared spaces can exist simultaneously in the application domain.

The Seine/MACE programming interface provides operators to initialize and access the shared spaces. These include *init*, *register*, *put*, *get*, and *rd*, as listed in Table 1. The Seine runtime is initialized using the *init* operator. The creation/destruction of a space does not require global synchronization and processors can individually and dynamically join or leave a space at runtime. A processor joins a space by registering its region of interaction. A processor leaves a space by de-registering the relevant region. When the last processor associated with a space de-registers, the space is destroyed. A processor inserts an object into the shared space using the *put* operator, which is functionally similar to *out* in Linda. A processor can retrieve an object using the *get* operator, which is functionally similar to *in* in Linda. The *get* operator is blocking and will wait until a matching object is written into the space. The *rd* operator is similar to *get*, except that unlike *get*, the object is not removed from the space. Arguments to these operators include a geometry descriptor to identify the space of interest and a tag to identify the object of interest.

**Table 1** Seine/MACE programming interface

| Operators | Function description | Linda |
|---|---|---|
| init | Uses a bootstrap mechanism to initialize the Seine runtime system | n/a |
| register | Registers a region with the Seine framework. Based on the geometric descriptor registered, a reference to an existing space or a newly created space is returned | n/a |
| put | Inserts an object into the shared space | out |
| get | Removes an object from the shared space. The *get* operator is blocking | in |
| rd | Copies an object from the shared space without removing it from the space. Multiple *rd* can be simultaneously invoked on an object | rd |

The implementation of a Seine/MACE framework complements existing interaction frameworks (e.g., MPI, OpenMP) and provides scalable geometry-based shared spaces for dynamic runtime coordination and localized communication. This framework uses the Hilbert Space Filling Curve, a locality preserving recursive mapping from a multi-dimensional coordinate space to a 1D index space, to construct a distributed directory structure that enables efficient geometric region registration and lookup of objects in the shared space. An experimental evaluation on up to 512 processors demonstrates both scalability and low operational overheads. Details of the implementation as well as experimental evaluation of Seine/MACE can be found (L. Zhang and M. Parashar, submitted).

### 3.3 Coupling flow, geomechanics and seismics

Flow, mechanics, and seismics are all coupled in the simulation of subsurface processes: a depletion or injection of fluids will change the pressure of a reservoir, and may also affect the mechanical properties of the rock matrix. These changes in turn will lead to a deformation of the reservoir, which in turn has an effect on fluid pressures. Finally, modified rock properties and a different geometry affect seismic reflections, wave amplitudes and two-way times which can be turned to visualize some of the subsurface changes using seismic imaging.

Within IPARS, fluid flow is described using single- or multiphase flow equations. However, in order to couple flow, geomechanics, and seismics, we need a relationship describing the correspondence between flow and mechanical properties. We will briefly outline such description in the following subsections.

*Petrophysical model* The purpose of fluid substitution is to simulate the effect of changes in the reservoir fluid properties on the isotropic elastic parameters. This analysis is usually accomplished by the use of Biot–Gassman theory. Applications include: time-lapse feasibility studies; prediction of amplitude and AVO (amplitude vs. offset) anomalies; and invasion corrections for better synthetic seismograms.

The Biot–Gassman theory describes the seismic velocity changes resulting from changes in pore-fluid saturations. The theory is mainly supported by the dependence that seismic velocities have with respect to saturated, dry, fluid and rock matrix bulk modulus, and

shear modulus [58]. The moduli are used to calculate elastic stiffness which defines wave propagation velocities. Other rock properties include porosity, shale volume, and grain density. Rock properties can be obtained from well logs, laboratory measurements of core properties, and correlations.

*Geomechanics model* The effects of geomechanics on seismic arrival changes have been observed in both numerical calculations and time-lapse (4D) seismic field monitoring of reservoirs undergoing depletion. For strongly stress-sensitive formations, reservoir characterization requires the integration of seismic surveillance and geomechanics analysis.

Different coupling methods for flow and geomechanics can be categorized as decoupled, explicitly coupled, iteratively coupled and fully coupled. Dean et al. [59] compared different coupling techniques in terms of efficiency and accuracy. Their numerical results indicated that the iterative method could be as accurate as a fully coupled scheme if a sufficiently tight tolerance is specified. For most reservoir compaction/subsidence problems it is more efficient than the fully coupled scheme, even though it takes more Newton iterations to converge. In [60], Gai demonstrated that iterative coupling may be viewed as a special case of the fully coupled method, thus it is unconditionally stable and does not have the time-step constraint as the explicit method does.

In the iterative coupling technique, the diffusion and elasticity operator are separated first by operator splitting. The decoupled equations are then solved sequentially at each nonlinear iteration as shown in Fig. 6. First the flow model solves the mass balance equations for pressure and concentrations by neglecting rock deformation effects. Then the geomechanics model uses the updated pressure and concentrations to compute displacements and stresses. The current iteration is terminated by updating the porosity according to a fluid fraction equation that depends on the computed pressures and fluid velocities The flow model will take the new porosity values and start another nonlinear iteration. Iteration continues until a given tolerance on residuals and pore volume is satisfied.

The effects of geomechanics on seismic arrival changes have been observed in both numerical calculations and time-lapse (4D) seismic field monitoring of reservoirs undergoing depletion [61–63]. The measured time-shifts are mainly caused by stress redistributions in the pay-zone and its surroundings as a result of reservoir compaction. To account for the effects of stress changes on seismic response, geomechanics studies need to be integrated into 4D seismic interpretations for strongly stress-sensitive formations.

*Seismic model* Seismic modeling is carried out using FDPSV and PWAVE3D codes. FDPSV is a time domain explicit staggered grid finite difference code that solves a first-order stress–displacement system assuming linear elasticity. The algorithm is very general and is valid for generally heterogeneous isotropic media. On the other hand, PWAVE3D is a fast algorithm that works in frequency wave number space. Here the medium is split into two parts. The background is assumed to be 1D to which perturbations are applied to approximate 3D variations.

Solving the flow model equations using the petrophysical relations and plugging in the corresponding seismic velocities to either FDPSV or PWAVE3D, we can compute the effect of changes in flow properties on seismic properties. We do so in Fig. 7: the top panels show P-wave velocities for an oil reservoir into which gas is injected at the left; obviously, the gas extends at the top of the oil reservoir towards the right, reducing the wave velocities in those areas where the gas concentrations are highest. At each time, we can use a seismic modeling code to predict the seismic signature of the reservoir (bottom row at different resolution levels). The effects of the changes in the reservoir are clearly visible in the seismic predictions. The incorporation of the geomechanics model into this computation add further prediction capabilities with respect to changes in the pore volume. The possibility to predict and monitor such changes using seismics in oil reservoirs that are currently in production, as well as the ability to interpret the changes in seismic signatures, is an important aspect of current research in geophysics and petroleum engineering.

The integration of flow, petrophysics, geomechanics and seismics models is key to achieve a more efficient and robust decisions as it was already depicted in Fig. 4.

Using these relations, we can compute the effect of changes in flow properties on seismic properties. We do so in Fig. 7: the top panels show P-wave velocities for an oil reservoir into which gas is injected at the left; obviously, the gas extends at the top of the oil reservoir towards the right, reducing the wave velocities in those areas where the gas concentrations are highest. At each time, we can use a seismic modeling code to predict the seismic signature of the reservoir (bottom row). The effects of the changes in the reservoir are clearly visible in the seismic predictions. The possibility to predict and monitor such changes using seismics in oil reservoirs that are currently in production, as well as the ability to interpret the changes in seismic signatures, is an important aspect of current research in geophysics and petroleum engineering.

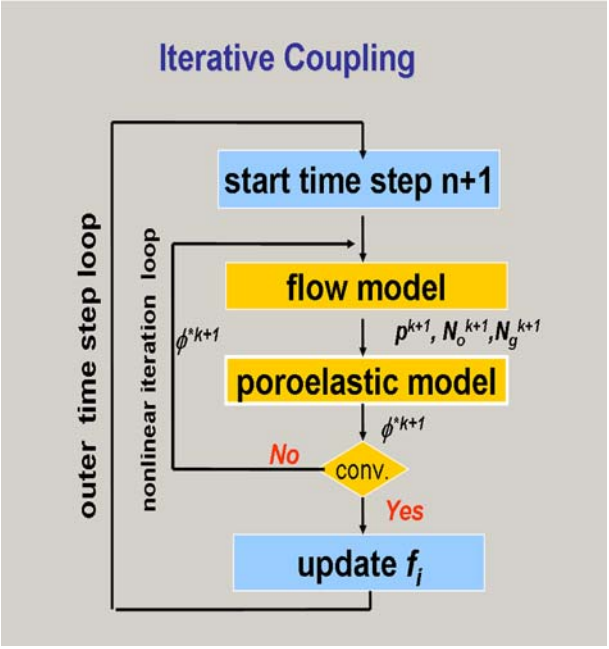


**Fig. 6** Iterative coupling of reservoir flow and geomechanics

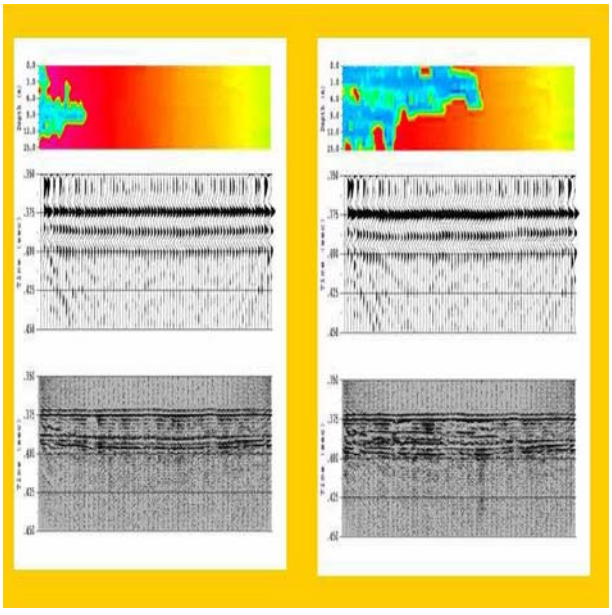

**Fig. 7** $V_p$ and the corresponding seismic response after 100 days (*left*) and 400 days (*right*) of flow simulation (*top*). Corresponding synthetic seismograms (*bottom*) at different resolution levels

### 3.4 Optimization algorithms

The DDMSF supports a family of different optimization algorithms. In [64] some of the authors describe experiences in comparing different approaches for the optimal well placement problem. The two algorithms we describe here and their extensions and hybridization with other algorithms open a promising avenue of research for large-scale applications.

*Simultaneous perturbation stochastic approximation* This method [29] is a random-direction version of the Kiefer–Wolfowitz algorithm. At each iteration, we simultaneously perturb all $N$ components of the present iterate by generating $N$ independent and identically distributed (i.i.d.) symmetric random variables (commonly) following a Bernoulli (i.e. $\pm\Delta x$) or pseudo-Bernoulli distribution. The gradient of the objective function is the estimate to be the finite difference approximation to the derivative in the direction of this perturbation. Therefore, the algorithm requires only two parallel function evaluations, i.e. simulations in our case, per iteration. A step in the descent direction is taken with a step length that is given by the product of the approximate value of the gradient and a factor that decreases with successive iterations.

Besides its efficiency, the SPSA algorithm is appealing since it works as a variant of the nonlinear steepest descent method if the objective function is deterministic, but is equally effective as a stochastic algorithm if the objective function contains noise. It can even be converted to a global optimization algorithm by cautious injection of noise into the objective function. Recently, SPSA has been topic of interest in several soft computing applications such as neural networks, see e.g., [65, 66]. Grid computing implementations for reservoir optimization and management have been reported in [22, 23, 25].

*Very fast simulated annealing* This algorithm shares the property of other stochastic approximation algorithms in relying only on function evaluations. Simulated annealing attempts to mathematically capture the cooling process of a material by allowing random changes to the optimization parameters if this reduces the energy (objective function) of the system. While the temperature is high, changes that increase the energy are also likely to be accepted, but as the system cools (anneals), such changes are less and less likely to be accepted.

Standard simulated annealing randomly samples the entire search space and moves to a new point if either the function value is lower there; or, if it is higher, the new point is accepted with a certain probability that decreases over time (controlled by the temperature decreasing with time) and by the amount by which the new function value would be worse than the old one. On the other hand, VFSA also restricts the search space over time, by increasing the probability for sampling points closer rather

than farther away from the present point as the temperature decreases. The first of these two parts of VFSA ensures that as iterations proceed we are more likely to accept only steps that reduce the objective function, whereas the second part effectively limits the search to the local neighborhood of our present iterate as we approach convergence. The rates by which these two probabilities change are controlled by the ''schedule'' for the temperature parameter; this schedule is used for tuning the algorithm. VFSA has been used successfully in several geophysical inversion applications [27, 67]. Alternative description of the algorithm can be found in [26].

Both SPSA and VFSA are gradient-free, non-intrusive optimization algorithms. This feature allows us to achieve both modularity and flexibility of using them interchangeably in a black-box fashion. Moreover, they are both suitable for performing a systematic and dense sampling on those regions that are most likely to lead a global optimal solution. Construction of surrogate models out of this sampling (i.e., local response surface metamodels) are convenient for eventually replacing the behavior of the simulation model by a cheaper computational model. This is key for generating faster responses for decision making and uncertainty analysis in our DDMSF approach.

## 4 Autonomic computational engine and grid middleware substrate

Emerging knowledge-based and dynamic data-driven geosystem management and control applications, such as the applications described in this paper, combine computations, experiments, observations, and real-time data, and are highly heterogeneous and dynamic in their scales, behaviors, couplings and interactions. Furthermore, the underlying enabling computational and information grid is similarly heterogeneous and dynamic, globally aggregating large numbers of independent computing and communication resources, data stores and sensor networks. Together, these characteristics result in complexities and challenges that require a fundamentally different approach to how the applications are formulated, developed and managed—one in which applications are capable of managing and adapting themselves in accordance with high-level rules from the experts based on their state, the available information and their execution context [68]. AutoMate [44], an autonomic computational engine for geosystem management and control, investigates conceptual models and implementation architectures to address these challenges and enable the development and execution of such self-managing grid applications. Key research components of AutoMate are described below.

### 4.1 Autonomic computational engine

The simulations targeted by this research and the phenomena they model are inherently dynamic and heterogeneous (in time, space, and state). Further, they employ advanced adaptive solution techniques, such as multi-block and adaptive mesh refinement. As a result, the appropriate behaviors of application elements and their compositions can no longer be statically defined—they depend on the application state, current information and the execution context, and are know only at runtime. As a result, applications must be able to detect and dynamically respond during execution to changes in both the execution environment and application state. This requirement suggests that (1) the applications should be composed from discrete, self-managing components that incorporate separate specifications for all of functional, non-functional and interaction–coordination behaviors, (2) the specifications of computational (functional) behaviors, interaction and coordination behaviors and non-functional behaviors (e.g., performance, fault detection and recovery, etc.) should be separated so that their combinations are composedly, and (3) the interface definitions of these components should be separated from their implementations to enable heterogeneous components to interact and to enable dynamic selection of components.

The autonomic grid-based computational engine supports self-managing and optimizing, dynamically adaptive geosytem simulations, using sophisticated numerical techniques based on multiblock grids, adaptive mesh refinement and multigrid. The key component is the *Accord* programming framework [69, 70] that enables the definition of autonomic components and the dynamic composition, management and optimization of these components using externally defined rules and constraints. Autonomic components in *Accord* export three programmable ports: a *functional port* defining the functionalities provided or used by the component, a *control port* exposing sensors and actuators for external monitoring and steering the component, and an *operational port* encapsulating rules for managing runtime behaviors of the component. A rule agent (possibly embedded) evaluates and executes rules to dynamically (and consistently) change the computational behaviors of components in response to current context and/or external events and

injected rules/constraints [71]. *Accord* builds on and complement emerging components/service based programming paradigms. Current implementations of Accord include:

- An object based prototype of Accord, named DIOS++ [72], implements autonomic elements as autonomic objects by associating objects with sensors, actuators and rule agents, and providing a runtime hierarchical infrastructure consisting of rule agents and rule engines for the rule-based autonomic monitoring and control of parallel an d distributed applications.
- A component based prototype of Accord, named Accord-CCA [73], based on the DoE CCA and the Ccaffeine framework in the context of component-based high-performance scientific applications. This prototype extends CCA components to autonomic components by associating them with control and operation ports and component managers, and provides a runtime infrastructure of component managers and composition managers for rule-based component adaptation and dynamic replacement of components.
- A service based prototype of Accord, named Accord-WS [74], based on the WS-Resource specifications, the Web service specifications, and the Axis framework. Autonomic elements are implemented as autonomic service by extending traditional WS-Resources with service managers for rule-based management of runtime behaviors and interactions with other autonomic services, and coordination agents for programmable communications. A distributed runtime infrastructure is investigated to enable decentralized and dynamic compositions of autonomic services.

Accord is currently being used to enable autonomic simulations in subsurface modeling, combustion and other areas [22, 24, 73, 74]. Further, the prototype implementations interface with advanced feature-based visualization techniques to enable both interactive [75] as well as rule-based automated [76] visualization and feature-tracking.

The *autonomic runtime application management* substrate provides policies and mechanisms for both ''system sensitive'' and ''application sensitive'' runtime adaptations to manage the heterogeneity and dynamism of the applications as well as grid environments. The former are driven by the current system state and system performance predictions while the latter are based on the current state of application. The overall goal is to maximize solution quality and computational efficiency for the given set of available resources and their current state. Prototype implementations [77] have demonstrated both the feasibility and the effectiveness of the autonomic runtime substrate in managing the complexity, heterogeneity and dynamism of grid environments.

### 4.2 Autonomic grid middleware

The content-based grid middleware supports autonomic application behaviors and interactions, and to enable simulation components, sensors/actuators, data archives and grid resources and services to seamlessly interact as peers. For example, simulation components interact with grid services to dynamically obtain necessary resources, detect current resource states, and negotiate required quality of service. Further, the data necessary for simulation is usually sparse and incomplete. Therefore, the simulation components must interact with one another and with data archives and real-time sensors to enable a better characterization and understanding of the subsurface model. The simulation components may interact with other services on the grid, for example, with optimization services such as the VFSA or SPSA algorithms to optimize a given objective function. Finally, the experts (scientist, engineers, and managers) collaboratively access, monitor, and steer the simulations and data at runtime to drive the discovery process. The processes described above must be autonomic in that the behaviors of the interacting elements and their interactions must be dynamically orchestrated using high-level polices defined only at runtime. These polices will enable the elements involved to automatically detect sub-optimal behaviors at runtime and opportunistically orchestrate interactions to correct this behavior.

A key component of the middleware is Meteor [78], a scalable content-based middleware infrastructure that provides services for content routing, content discovery and associative interactions. The Meteor stack consists of three key components: (1) a self-organizing content overlay, (2) a content-based routing engine and discovery service (Squid), and (3) the associative rendezvous messaging substrate (ARMS). The Meteor overlay is composed of peer nodes, which may be any node on the grid (e.g., gateways, access points, message relay nodes, servers or end-user computers). These nodes can join or leave the network at any time. The overlay topology is based on standard structured overlays. The content overlay provides a single operation, *lookup(identifier)*, which requires an exact content identifier (e.g., name). Given an identifier, this operation locates the peer node where the content should be stored.

Squid [79] is the Meteor content-based routing engine and decentralized information discovery service. It support flexible content-based routing and complex queries containing partial keywords, wildcards, and ranges, and guarantees that all existing data elements that match a query will be found. The key innovation of Squid is the use of a locality preserving and dimension reducing indexing scheme, based on the Hilbert Space Filling Curve, which effectively maps the multidimensional information space to the peer identifier space. Squid effectively maps complex queries consisting of keyword tuples (multiple keywords, partial keywords, wildcards, and ranges) onto clusters of identifiers, and guarantees that all peers responsible for identifiers in these clusters will be located. Keywords can be common words or values of globally defined attributes, depending on the nature of the application that uses Squid, and are based on common ontologies and taxonomies.

The ARMS layer [78] implements the associative rendezvous (AR) interaction paradigm. AR is a paradigm for content-based decoupled interactions with programmable reactive behaviors. Rendezvous-based interactions provide a mechanism for decoupling senders and receivers, in both space and time. Such decoupled asynchronous interactions are naturally suited for large, distributed, and highly dynamic systems such as pervasive grid environments. AR extends the conventional name/identifier-based rendezvous in two ways. First, it uses flexible combinations of keywords (i.e, keyword, partial keyword, wildcards and ranges) from a semantic information space, instead of opaque identifiers (names, addresses) that have to be globally known. Interactions are based on content described by these keywords. Second, it enables the reactive behaviors at the rendezvous points to be encapsulated within messages increasing flexibility and enabling multiple interaction semantics (e.g., broadcast multicast, notification, publisher/subscriber, mobility, etc.).

### 4.3 The discover collaboratory

The utility and cost-effectiveness of large-scale scientific and engineering process can be greatly increased by transforming the traditional batch applications into more interactive and collaborative ones. Closing the loop between the user and the applications enables experts to drive the discovery process by observing intermediate results, by changing parameters to lead the simulation to more interesting domains, play what-if games, detect and correct unstable situations, and terminate uninteresting runs early. Furthermore, the increased complexity and multi-disciplinary nature of these simulations necessitates a collaborative effort among multiple, usually geographically distributed scientists/engineers. As a result, collaboration-enabling tools are critical for the applications processes.

The overall objective of the Discover computational collaboratory [80, 81] is to realize a collaborative problem solving environment that enables geographically distributed scientists and engineers to collaboratively monitor, interact with, and control high performance applications in a truly pervasive manner. Its goal is to transform high-performance simulations into true modalities for research and instruction. Key features of Discover include detachable, pervasive (web based) collaborative portals for interaction and control, mechanisms for web-based runtime visualization, scalable interaction and collaboration servers networks that reliably provide uniform access to remote distributed applications, and security, authentication, and access control mechanisms that guarantee authorized access to applications

## 5 Data management and data processing support

In order to enable dynamic data driven approaches in simulation studies, we need support for gleaning and extracting information from results of complex numerical models, which are large, multi-scale in time and space, and heterogeneous, and from data gathered by field sensors. These datasets are stored on large-scale storage systems, consisting of clusters of disk-based storage nodes, and can be distributed across multiple storage nodes in a grid environment. There are some recent efforts to develop grid services [82, 83] and Web services [84] implementations of database technologies [85]. Raman et al. [86] discusses a number of *virtualization* services to make data management and access transparent to grid applications. These services provide support for dynamic discovery of data sources and collaboration. Bell et al. [87] develop uniform web services interfaces for relational databases. The goal is to address interoperability between database systems at multiple organizations. Smith et al. [88] address the distributed execution of queries in a grid environment. In addition to supporting interoperability among databases, there is a need for tools that support storage, management, and querying of very large and distributed scientific datasets. These datasets are oftentimes stored in a set of distributed files. In this work, we develop data handling techniques and middleware frameworks which form the GeoDAM. GeoDAM encapsulates methods, optimizations, and tools

in a distributed service-based software platform to support large-scale data management, access, and analysis and to harness the disk and memory capacity and I/O bandwidth of very large disk-based storage systems. It builds on two middleware components DataCutter [5] and STORM [89, 90] that are designed to provide high performance support for data subsetting and distributed data processing.

### 5.1 STORM and DataCutter

STORM is a service-oriented middleware that supports data select and data transfer operations on scientific datasets, stored in distributed, flat files, through an object-relational database model. In STORM, data subsetting is done based on attribute values or ranges of values, and can involve user-defined filtering operations. STORM services provide support to create a view of data files in the form of virtual tables using application specific *extraction* objects. STORM is structured as a suite of loosely coupled services: (1) the *query service*, where clients submit queries to the database middleware; (2) the *meta-data service*, that maintains information about datasets, and indexes and user-defined filters associated with the datasets; (3) the *indexing service* that encapsulates indexes for a dataset; (4) the *filtering service* that is responsible for execution of user-defined filters; (5) the *partition generation service* that allows an application developer to implement the data distribution scheme employed in the client program at the server; and (6) the *data mover* service which is responsible for transferring selected data elements to destination processors based on the partitioning description generated by (5). STORM implements several optimizations to reduce the execution time of queries. These optimizations include (1) ability to execute a workflow through distributed filtering operations, and (2) execution of parallelized data transfer. Both data and task parallelism can be employed to execute filtering operations in a distributed manner. If a select expression contains multiple user-defined filters, a network of filters can be formed and executed on a distributed collection of machines. Data is transferred from multiple data sources to multiple destination processors in parallel by STORM data mover components.

DataCutter is a middleware system designed to support processing of large datasets in a distributed environment. A DataCutter application consists of a network of interacting application-specific components, called *filters*, one or more *filter groups*. Filters are connected through logical streams and collectively realize the processing structure of the application. A logical stream denotes a uni-directional data flow from one filter (i.e., the producer) to another (i.e., the consumer). DataCutter allows for combined use of task-parallelism, data-parallelism, and pipelining for reducing execution time of data processing and analysis applications. Using the DataCutter and STORM components, the GeoDAM system provides the following functionality.

### 5.2 Data virtualization and data subsetting

A major barrier to effective utilization of distributed collections of data sets is that the types and formats of datasets vary widely. Most scientific datasets are stored in files with different formats, making it difficult to search for and extract the data of interest. In order to provide support for data querying and manipulation on such datasets, a level of abstraction is needed that will separate application specific characteristics from processes that query and analyze the data. These abstractions are *virtual tables* based on object-relational database models, *select queries*, and *distributed data descriptors*. A dataset can be viewed as a table. The rows of the table correspond to data elements, each consisting of a set of attributes and attribute values. The data analysis program that process the data can be a parallel program implemented using a distributed-memory programming paradigm. The *distributed data descriptor* abstraction is utilized to specify how data elements selected from the database are to be distributed across processing nodes. This functionality is supported by the STORM component. The metadata and filtering services of STORM implement the support for on-the-fly generation of virtual tables on top of the data files of the dataset. The query and partition generation services provide the support for select queries and distributed data descriptors.

### 5.3 Support for management and processing of very large (100 TB scale) datasets at data centers

With the help of inexpensive disk-based storage, we are seeing the emergence of *data centers* with large-scale mass storage platforms. Because of cost-performance considerations and the need to support wide range of applications, these mass storage platforms are made up of multiple levels of storage with varying capacity/bandwidth (from larger, slower disk pools to smaller, faster disks to memory on compute cluster) and distance from compute resources.

Several optimizations can be applied when storing and accessing very large terabyte-scale datasets to

minimize the time spent for retrieving the data of interest. (1) *Data declustering*. Efficient access to data depends on how well the data has been distributed across storage units, both within a storage level and across levels. The declustering of the dataset should be done in such a way that a request to read a portion of the dataset would be served by as many storage units as possible and I/O load is distributed based on the I/O bandwidth of the storage units. (2) *Data indexing*. The datasets can be indexed to speed up searches for data elements that intersect a given query. When dataset collections and datasets reach several terabytes or petabytes in size, the index for the entire dataset can be extremely large. As a result, it may be very expensive to manage the index and search for data elements that satisfy a query using a single index file. In that case, a hierarchical multi-level indexing scheme may provide an efficient solution. (3) *Data caching and replication*. Multiple queries are expected, the data can be cached on faster disks or in memory so that it can be accessed much faster next time it is requested. To use aggregate system memory effectively, the data to be cached should be distributed across the nodes in the system, both to provide a large data cache and to achieve parallelism when data is accessed. Another optimization would be partial replication of input datasets. If most of the queries to a dataset collection accesses a common subset of data, that portion can be extracted, redistributed across the disks at the same or a higher level storage to minimize search and data extraction overheads. The support for data declustering and indexing is provided by the meta-data and indexing services of STORM. We have also implemented prototype support for data replication and caching in STORM.

### 5.4 Distributed storage and processing of data on storage islands

It is reasonable to anticipate that the collection of datasets comprising a simulation study will be distributed across a wide-area network, since computational and storage demands compel the use of multiple supercomputers. A similar expectation is true for field measured data. PC clusters built from low-cost, commodity items are increasingly becoming widely used. With fast CPUs and high-speed interconnects, they provide cost-effective *compute nodes* for computation intensive applications. With high-capacity, commodity disks, they create *active storage nodes* that enhance a scientist's ability to store large-scale scientific data. Active storage clusters can be employed as *storage islands*, which maintain data from nearby data sources. For example, an institution may deploy a medium-size cluster that stores datasets generated by large simulations executed at a nearby supercomputer facility.

A storage island can also be set up near field sensors that feed data into the storage system. Such configurations have two main advantages. First, it reduces cost by directing low-bandwidth links to nearby storage and staging platform, which may have higher-bandwidth connection to other systems. Second, with proper middleware systems, the data islands can be utilized as on-demand dynamic data generation platforms. That is, we can run codes to integrate data from different sensors or different simulations, filter out data, or on-demand produce an aggregate data production on the data island platform before transferring data to data centers for data assimilation.

This functionality is supported by the DataCutter component of GeoDAM. DataCutter allows an application developer to implement data processing filters and execute them storage islands near data sources.

## 6 Results

### 6.1 Evaluations Seine/MACE interaction framework

The performance of the Seine/MACE framework has been evaluated using a parallel multi-block oil reservoir simulation consisting of six 3D grid blocks and five 2D mortar-grids at the interfaces of the blocks. The geometry-based shared spaces were used to share data on mortar grid objects at the interfaces between blocks. Note that the application used Seine/MACE for the couplings between the blocks and MPI for all other communications.

The experiments were conducted on a 64 node Beowulf cluster. The Beowulf cluster has 64 Linux-based computers connected by 100 Mbps full-duplex switches. Each node has an Intel(R) Pentium-4 1.70 GHz CPU with 512 MB RAM and runs Linux 2.4.20-8 (kernel version). The experiments consist of measuring the time for *register*, *get* and *put* operations for a range of system sizes, from 8 to 64 processors. In each case, the time for each operation was averaged across the processors. The results are plotted in Fig. 8. Note that the metric used here is the overall average cost rather than the average cost per unit size.

As seen in the figures, the system startup time increases as system size increases, while the times for *register*, *get* and *put* operations decrease. The increase in startup time is due to the client-server nature of bootstrapping in the current implementation. As the

system size increases, the average size of objects and corresponding regions decreases. The reason is that as the system size increases, each block will be mapped to a larger number of processors and the size of the sub-block (and corresponding shared interface) at each processor will be smaller. Since the size of the registered region is the dominant factor contributing to the cost of an operation, the overall cost decreases as the system size increases.

### 6.2 Autonomic optimal production

The optimal well location on the grid is illustrated in Fig. 9. Discover provides the portal for users to interact with different optimization services and the IPARS factory triggers different parallel instances of IPARS corresponding to different well configurations. Previous computed configurations are checked in the database in order to reduce the computations and guide the optimal search to other unexplored regions. Users can interact among themselves and the application to analyze the progress towards the optimal solution.

The optimization of well locations using the VFSA and SPSA optimization algorithms for two different scenarios are presented in Fig. 10. The goal is to maximize profits for a given economic revenue objective function. The well positions plots (top and bottom of Fig. 10) show the oil field and the positions of the wells. Black circles represent fixed production wells and a gray square at the bottom of the plot is a fixed injection well. The plots also show the sequence of guesses for the position of the other injection well returned by the optimization service (shown by the lines connecting the light squares), and the corresponding normalized cost value (top and bottom of Fig. 10). Further details can be found [22, 24].

### 6.3 Uncertainty management

Another important aspect of autonomic optimization is to capture the effects of uncertainty: in general, we may have some information about the properties of an oil reservoir, but due to its largely inaccessible nature, our picture may be incomplete. Recent approaches to cope with this lack of information center around simulating with several different reservoir models, all of which fit the knowledge we have, but can be considered different realizations of a probability space representation of what we know about the reservoir.

This creates several opportunities for parallelism, all of which should be exploited for efficient solution of the problem:

- Simulations for different reservoir models are independent of each other; the only information that is important to us are ensemble averages over our set of stochastic realizations of our reservoir model.

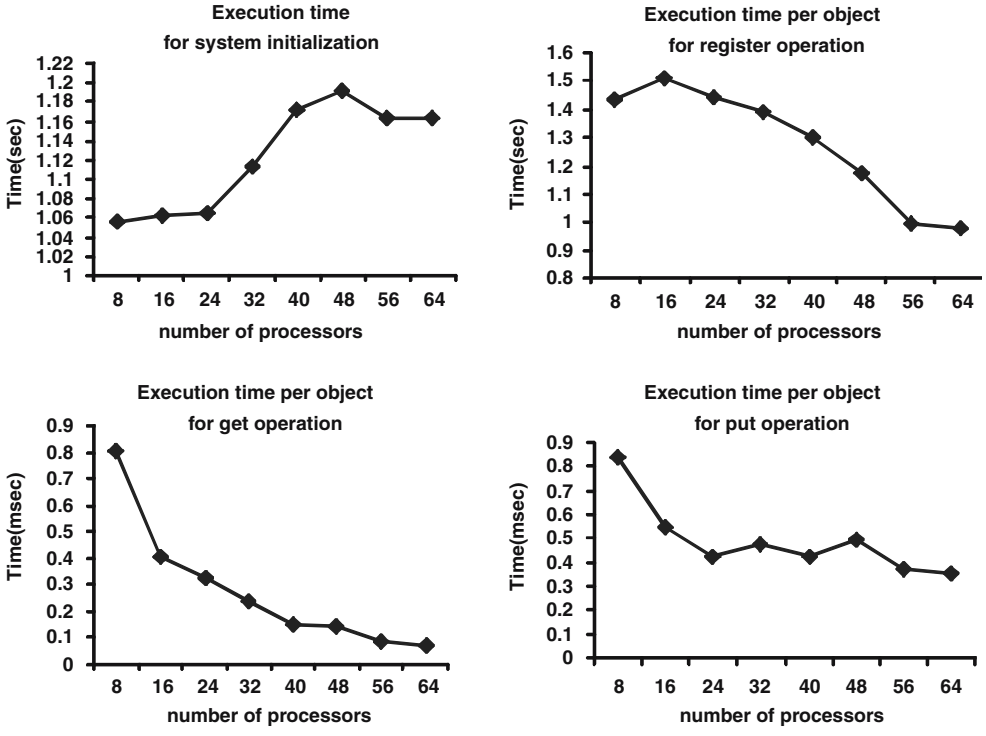


**Fig. 8** Average execution time for system initialization and per object *register*, *get* and *put* operations on the 64-node Beowulf cluster

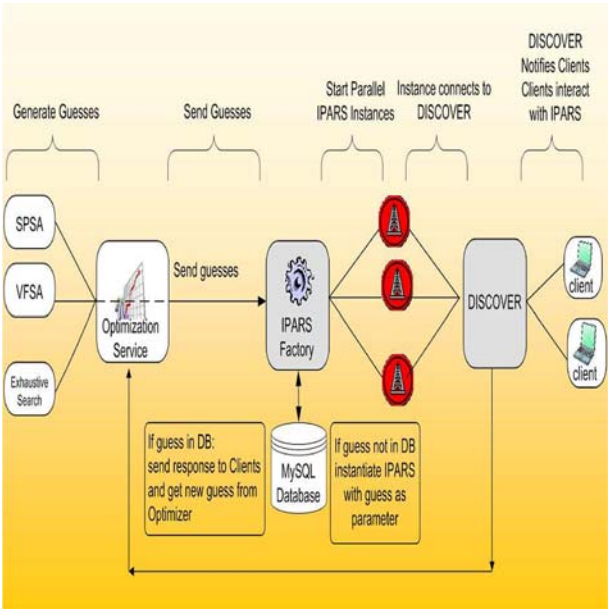


**Fig. 9** The autonomic oil reservoir optimization system on the grid

- Different initial guesses: since optimizers may find different solutions for non-convex problems when using different starting positions, we run several instances of the optimization algorithm in parallel, in order to compare the best solutions found by each instance. These runs are completely independent, however; we are only interested in comparing the final result of each optimization run.
- Different optimizers: some optimization algorithms get stuck more easily in local minima; if the properties of the solution surface as described by the objective function are unknown, it may be prudent to run several different optimization algorithms in parallel, and to compare the quality of the solution each of them finds. These runs, again, are independent of each other.
- Finally, IPARS itself can run in parallel. Each IPARS run may therefore be distributed across a number of machines, which will be in tight contact during the solution of a problem.

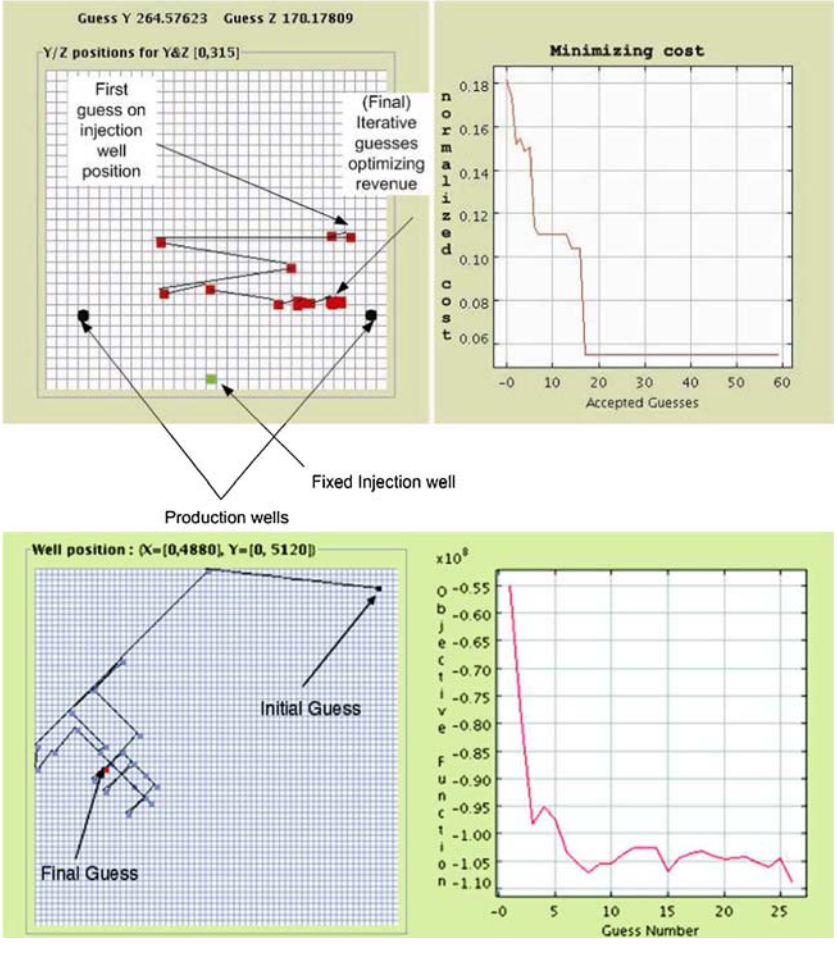


**Fig. 10** Convergence history for the optimal well placement in the grid using VFSA algorithm (*top*) and SPSA algorithm (*bottom*) [91]

All these levels of parallelism can be used to distribute extremely large numbers of simulations across tightly or loosely coupled clusters of machines. Using grid technology, we schedule the next IPARS simulation (as requested by a particular instance of a particular optimization algorithm started from a certain initial guess) to run on available resources, wait for its termination, evaluate the result and possible pass the output on to the next program in line, such as a seismic simulator. If all this is done, we schedule the next job. Figure 11 illustrates how a three level parallelism can be exploited in the grid.

### 6.4 Flow/Seismic data management

Figure 12 shows the performance of the STORM component of GeoDAM for querying and subsetting seismic datasets. The performance numbers were obtained on a 30TB seismic dataset generated by simulations and stored on a 16-node disk-based cluster storage system, with 4.3 GB/s peak application-level bandwidth, at the Ohio Supercomputer Center. As seen from the figure, we can achieve close to 3.5 GB/s (about 75% of the peak bandwidth) bandwidth through runtime optimizations (such as distributed I/O, distributed filtering, multi-threading) implemented by STORM.

We also evaluated the support for partial replication using a large dataset with characteristics similar to those created by oil reservoir management applications. All of the experiments were carried out on eight nodes in a Linux cluster where each node has a PIII 933 MHz CPU, 512 MB main memory, and three 100 GB IDE disks. The nodes are inter-connected via a Switched Fast Ethernet. We generated a 0.35 TB size dataset, with the same domain partitioning as the simulator code, so that we can manage the distribution of attributes for a more controlled experimental environment. Each grid point is stored as a tuple and each tuple consists of 21 attributes. Sixteen time steps worth of data is created using a grid of size 1,024 $\times$ 1,024 $\times$ 256. The data is partitioned into 8 $\times$ 8 $\times$ 256 $\times$ 1 size chunks on $X$, $Y$, $Z$ and $TIME$ dimensions (attributes). Each chunk is roughly 1.3 MB in size. The metadata associated with a chunk includes lower and upper bounds of each attribute along with a (*filename, offset, size*) triple that is required to retrieve it. The chunks were declustered across the data nodes. Each node maintains a local index of its chunks' metadata. The index takes a query as input and returns the list of chunks whose bounds intersect with the range of the query. The attributes that we will focus on are *SOIL* (oil saturation), which has a uniform distribution in [0, 1], and the VX (oil velocity in $x$-dimension) attribute which has a standard normal distribution. Queries are sliding window queries,each of which is of the form $Q = [(l_X, l_Y, l_T, l_{\mathrm{SOIL}}, l_{\mathrm{VX}}):(h_X, h_Y, h_T, h_{\mathrm{SOIL}}, h_{\mathrm{VX}})]$.

Figure 13 shows the time taken to execute the queries. Performance of replication ratios 0.5 and 1.0 are shown in the above figures. The replicated data is partitioned along SOIL and VX dimensions using both uniform and recursive partitioning techniques. This

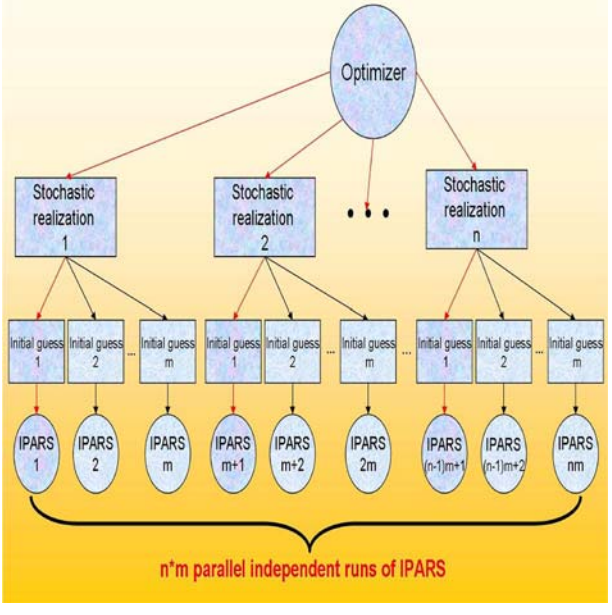


**Fig. 11** Uncertainty analysis leads to a three level of parallelism

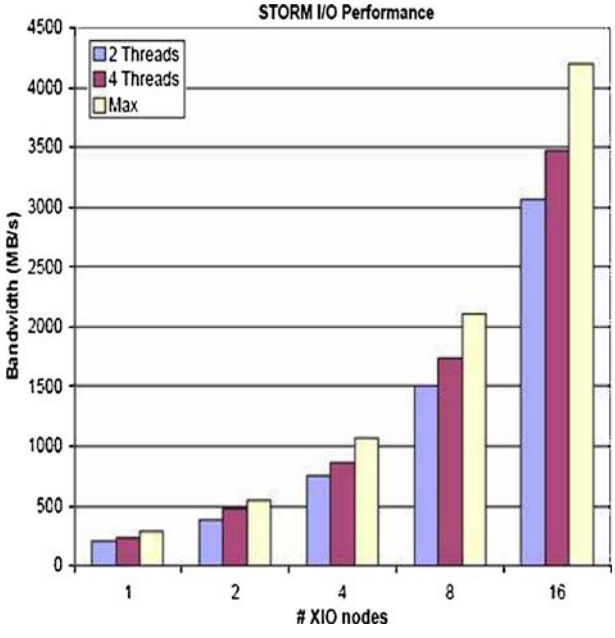


**Fig. 12** Querying seismic data using STORM

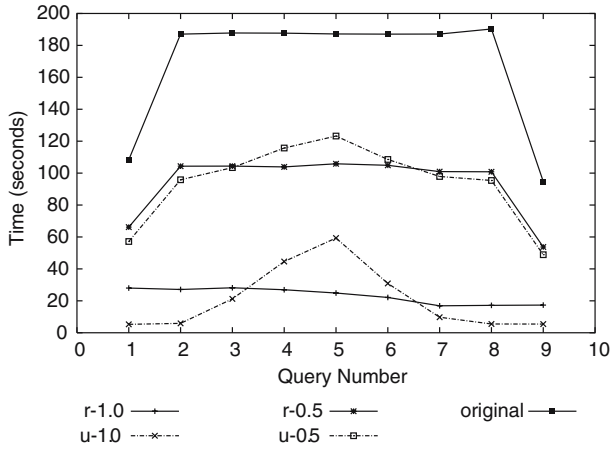


**Fig. 13** Query execution time with different replicas

decreases spurious I/O and improves query performance. We can see increased benefits for the sliding window queries as the replication ratio is increased.

## 7 Conclusions

Grid computing enables the development of large oil engineering applications to an unprecedented scale. The philosophy of ''on-demand'' availability of computational resources is a challenging topic of research for dealing with the different processes and scales governing the exploration and production phase of a reservoir.

The present paper has offered a broad overview of recent computational developments aiming at facilitating the incorporation of more complex processes, data, interaction and understanding of the oil reservoir. The advent of new sensor technology and computing power has established new and shorter scientific connections between different areas that have traditionally coexisted in an isolated fashion in the industry, such as reservoir simulation, geophysics, petrophysics and geomechanics.

We have shown how grid middleware and data management tools enable and support the computation of different physics, scales, algorithms towards reducing uncertainty, increasing the reliability of production decision-making and oil exploitation planing.

The present team believes that the development of more flexible and efficient grid environments would enable engineers and scientists to efficiently exploit this technology and significantly increase the understanding and control the oil reservoir studies.

**Acknowledgments** The authors want to thank the National Science Foundation (NSF) for its support under the ITR grant EIA-0121523/ EIA-0120934, grants #ACI-9619020 (UC Subcontract #10152408), #EIA-0121177, #ACI-0203846, #ACI-0130437, #ANI-0330612, #ACI-9982087, #CCF-0342615, #CNS-0406386, #CNS-0426241, #ACI-9984357, #EIA –0103674, #ANI-0335244, #CNS-0305495, #CNS-0426354 and #IIS-0430826, Lawrence Livermore National Laboratory under Grant #B517095 (UC Subcontract #10184497), and grants from Ohio Board of Regents BRTTC #BRTT02-0003.

## References

1. Foster I, Kesselman C (eds) (1999) Globus: a toolkit based grid architecture. Morgan Kaufman, San Francisco, pp 259–278
2. Frey J, Tannenbaum T, Foster I, Livny M, Tuecke S (2001) Condor-G: a computation management agent for multi-institutional grids. In: Proceedings of the 10th IEEE symposium on high performance distributed computing (HPDC10). IEEE Press, New York
3. Rajasekar A, Wan M, Moore R (2002) MySRB & SRB—components of a data grid. In: The 11th international symposium on high performance distributed computing (HPDC-11)
4. Wolski R, Spring N, Hayes J (1999) The Network Weather Service: a distributed resource performance forecasting service for metacomputing. J Future Gener Comput Syst 15(5–6):757–768
5. Beynon MD, Kurc T, Catalyurek U, Chang C, Sussman A, Saltz J (2001) Distributed processing of very large datasets with DataCutter. Parallel Comput 27(11):1457–1478
6. Grimshaw AS, Wulf WA, the Legion Team (1997) The legion vision of a worldwide virtual computer. Commun ACM 40(1):39–45
7. Allen G, Dramlitsch T, Foster I, Karonis N, Ripeanu M, Seidel E, Toonen B (2001) Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus. In: Proceedings of the ACM/IEEE SC1001 conference. ACM Press, New York
8. Common Component Architecture Forum. http://www.cca-forum.org
9. Allcock W, Chervenak A, Foster I, Kesselman C, Salisbury C, Tuecke S (2001) The DataGrid: towards an architecture for the distributed management and analysis of large scientific datasets. J Netw Comput Appl 23:187–200
10. Casanova H, Dongarra J (1998) Applying Netsolve's network-enabled server. IEEE Comput Sci Eng 5(3):57–67
11. Czajkowski K, Fitzgerald S, Foster I, Kesselman C (2001) Grid Information services for distributed resource sharing. In: 10th IEEE symposium on high performance distributed computing
12. Oldfield R, Kotz D (2001) Armada: a parallel file system for computational grids. In: Proceedings of CCGrid2001: IEEE international symposium on cluster computing and the grid. IEEE Computer Society Press, Brisbane, Australia
13. Sato M, Nakada H, Sekiguchi S, Matsuoka S, Nagashima U, Takagi H (1997) Ninf: a network based Information Library for a global world-wide computing infrastructure. In: Proceedings of HPCN'97 (LNCS-1225), pp 491–502
14. Thain D, Basney J, Son S, Livny M (2001) Kangaroo approach to data movement on the grid. In: Proceedings of the 10th IEEE symposium on high performance distributed computing (HPDC10)
15. Thain D, Bent J, Arpaci-Dusseau A, Arpaci-Dusseau R, Livny M (2001) Gathering at the well: creating communities for grid I/O. In: Proceedings of supercomputing 2001. Denver, CO, USA

16. Vazhkudai S, Tuecke S, Foster I (2001) Replica selection in the Globus data grid. In: International workshop on data models and databases on Clusters and the grid (DataGrid 2001). IEEE Computer Society Press, New York
17. Bevc D (2003) eBusiness and geophysics, vol 22. The Leading Edge, Provo, pp 53–53
18. Bevc D, Popovici M (2003) Integrated Internet collaboration, vol 22. The Leading Edge, Provo, pp 54–57
19. Fuller J, Fay J (2003) How the Internet is influencing today's E&P business, vol 22. The Leading Edge, Provo, pp 65–68
20. Hanley S (2003) The collaborative power of IT leads industry transformation, vol 22. The Leading Edge, Provo, pp 62–64
21. Karbarz F (2003) Grid computing for seismic processing, vol 22. The Leading Edge, Provo, pp 58–60
22. Bangerth W, Matossian V, Parashar M, Klie H, Wheeler M (2005) An autonomic reservoir framework for the stochastic optimization of well placement. Cluster Comput J Netw Softw Tools 8(4):255–269
23. Klie H, Bangerth W, Wheeler MF, Parashar M, Matossian V (2004) Parallel well location optimization using stochastic algorithms on the grid computational framework. In: 9th European conference on the mathematics of oil recovery (ECMOR). EAGE, August 30–September 2 2004
24. Matossian V, Bhat V, Parashar M, Peszynska M, Sen M, Stoffa P, Wheeler MF (2005) Autonomic oil reservoir optimization on the grid. Concur Comput Pract Exp 17(1):1–26
25. Parashar M, Klie H, Catalyurek U, Kurc T, Bangerth W, Matossian V, Saltz J, Wheeler MF (2005) Application of grid-enabled technologies for solving optimization problems in data-driven reservoir studies. J Future Gener Comput Syst Spec Issue Eng Auton Syst 21(1):19–26
26. Ingber L (1989) Very fast simulated reannealing. Math Comput Model 12:967–993
27. Sen M, Stoffa P (1995) Global optimization methods in geophysical inversion. Elsevier, Amsterdam
28. Spall JC (1992) Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Trans Autom Control 37:332–341
29. Spall JC (2003) Introduction to stochastic search and optimization: Estimation, simulation and control. Wiley, New Jersey
30. Gai X, Dean R, Wheeler MF, Liu R (2003) Coupled geomechanical and reservoir modeling on parallel computers. In: SPE 79700, proceedings of SPE reservoir symposium, Houston, TX
31. Lu Q, Peszyńska M, Wheeler MF (2001) A parallel multiblock black-oil model in multi-model Implementation. In: 2001 SPE reservoir simulation symposium, Houston, TX, SPE 66359
32. Minkoff S, Stone CM, Arguello JG, Bryant S, Eaton J, Peszynska M, Wheeler MF (1999) Staggered in time coupling of reservoir flow simulation and geomechanical deformation: Step 1—one-way coupling. In: 1999 SPE symposium on reservoir simulation, Houston, TX, SPE 51920
33. Parashar M, Wheeler JA, Pope G, Wang K, Wang P (1997) A new generation EOS compositional reservoir simulator. Part II: framework and multiprocessing. In: 14th SPE symposium on reservoir simulation, Dallas, TX, Society of Petroleum Engineers, pp 31–38
34. Peszyńska M, Lu Q, Wheeler MF (2000) Multiphysics coupling of codes. In: Bentley LR, Sykes JF, Brebbia CA, Gray WG, Pinder GF (eds) Computational methods in water resources. A. A. Balkema, Amsterdam, pp 175–182
35. Peszyńska M, Wheeler MF, Yotov I (2002) Mortar upscaling for multiphase flow in porous media. Comput Geosci 6(1):73–100
36. Wang P, Yotov I, Wheeler MF, Arbogast T, Dawson CN, Parashar M, Sepehrnoori K (1997) A new generation EOS compositional reservoir simulator. Part I: formulation and discretization. In: 14th SPE symposium on reservoir simulation, Dallas, TX, Society of Petroleum Engineers, pp 55–64
37. Wheeler MF (2002) Advanced techniques and algorithms for reservoir simulation, II: the multiblock approach in the integrated parallel accurate reservoir simulator (IPARS). In: Chadam J, Cunningham A, Ewing RE, Ortoleva P, Wheeler MF (eds) IMA volumes in mathematics and its applications, vol 131. Resource recovery, confinement, and remediation of environmental hazards. Springer, Berlin Heidelberg New York
38. Wheeler MF, Peszynska M (2002) Computational engineering and science methodologies for modeling and simulation of subsurface applications. Adv Water Resource 25(8):1147–1173
39. Wheeler MF, Wheeler JA, Peszyńska M (2000) A distributed computing portal for coupling multi-physics and multiple domains in porous media. In: Bentley LR, Sykes JF, Brebbia CA, Gray WG, Pinder GF (eds) Computational methods in water resources. A. A. Balkema, Amsterdam, pp 167–174
40. Arbogast T, Cowsar LC, Wheeler MF, Yotov I (2000) Mixed finite element methods on non-matching multiblock grids. SIAM J Numer Anal 37:1295–1315
41. Lu Q (2000) A parallel multi-block/multi-physics approach for multi-phase flow in porous media. PhD Thesis, University of Texas at Austin
42. Narayanan S, Catalyurek U, Kurc T, Zhang X, Saltz J (2003) Applying Database support for large scale data driven science in distributed environments. In: Proceedings of the 4th international workshop on grid computing (Grid 2003), Phoenix, Arizona, pp 141–148
43. Saltz J et al (2003) Driving scientific applications by data in distributed environments. In: Dynamic data driven application systems workshop, held jointly with ICCS 2003, Melbourne, Australia
44. Parashar M, Liu H, Li Z, Matossian V, Schmidt C, Zhang G, Hariri S (2006) AutoMate: enabling autonomic grid applications. Cluster Comput J Netw Softw Tools Appl Spec Issue Auton Comput 9(2):161–174
45. Lacroix S, Vassileski Y, Wheeler J, Wheeler M (2003) Iterative solution methods for modeling multiphase flow in porous media fully implicitly. SIAM J Sci Comput 25(3):905–926
46. Li J, Wheeler MF (2000) Uniform convergence and superconvergence of mixed finite element methods on anisotropically refined grids. SIAM J Numer Anal 38(3):770–798
47. Peszyńska M, Lu Q, Wheeler MF (1999) Coupling different numerical algorithms for two phase fluid flow. In: Whiteman JR (ed) MAFELAP Proceedings of mathematics of finite elements and applications. Brunel University, Uxbridge, UK, pp 205–214
48. Wheeler MF, Yotov I (1998) Physical and computational domain decompositions for modeling subsurface flows. In: Mandel J et al (eds) 10th international conference on domain decomposition methods, contemporary mathematics, vol 218. American Mathematical Society, pp 217–228
49. Wheeler MF, Arbogast T, Bryant S, Eaton J, Lu Q, Peszyńska M, Yotov I (1999) A parallel multiblock/multidomain approach to reservoir simulation. In: 15th SPE symposium on reservoir simulation, Houston, TX. Society of Petroleum Engineers. SPE 51884, pp 51–62
50. Wohlmuth BI (2000) A mortar finite element method using dual spaces for the Lagrange multiplier. SIAM J Numer Anal 38:989–1012

51. Yotov I (1996) Mixed finite element methods for flow in porous media. PhD Thesis, Rice University, Houston, TX. TR96-09, Dept. Comp. Appl. Math., Rice University and TICAM report 96-23, University of Texas at Austin
52. Wheeler MF, Yotov I (2005) A posteriori error estimates for the mortar mixed finite element method. SIAM J Numer Anal 43(3):1021–1042
53. Peszynska M, Sun S (2002) Reactive transport model coupled to multiphase flow models. In: Hassanizadeh SM, Schotting RJ, Gray WG, Pinder GF (eds) Computational methods in water resources. Elsevier, Amsterdam, pp 923–930
54. Snir M, Otto S, Huss-Lederman S, Walker D, Dongarra J (1996) MPI: the complete reference. MIT Press, New York
55. Carriero N, Gelernter D (1989) Linda in context. Commun ACM 32(0001–0782):444–458
56. Sterck H, Markel R, Pohl T, Rude U (2003) A lightweight Java taskspaces framework for scientific computing on compuatational grids. In: Proceedings of the 18th annual ACM symposium on applied computing, 1-58113-624-2, Melbourne, FL, USA, pp 1024–1030
57. Zhang L, Parashar M (2004) A dynamic geometry-based shared space interaction framework for parallel scientific applications. In: Proceedings of the 11th annual international conference on high performance computing (HiPC 2004), vol 3296. LNCS, Springer, Bangalore, pp 189–199
58. Bourbie T, O OC, Zinsner B (1987) Acoustics of porous media. IFP Publications, Paris
59. Dean R, Gai X, Stone C, Minkoff S (2003) A comparison of techniques for coupling porous flow and geomechanics. In: Tthe SPE reservoir simulation symposium. Houston, TX, SPE 79709
60. Gai X (2004) A coupled geomechanics and reservoir flow model on parallel computers. PhD Thesis, The University of Texas at Austin
61. Kenter C, van den Beukel A, Hatchell P, Maron K, Molenaar M (2004) Geomechanics and 4D: evaluation of reservoir characteristics from time-shifts in the overburden. In: Presented at Gulf Rocks 2004, Houston, TX, June 5–9. ARMA/NARMS 04-627
62. Minkoff S, Stone C, Arguello J, Bryant S, Eaton J, Peszyńska M, Wheeler M (1999) Coupled geomechanics and flow simulation for time-lapse seismic modeling. In: Expanded Abstracts, 1667–1670. Soc Expl Geophys
63. Molenaar M, Hatchell P, van den Beukel A (2004) 4D in-situ stress as a complementary tool for optimizing field management. In: Presented at Gulf Rocks 2004, Houston, TX, June 5–9. ARMA/NARMS 04-639
64. Bangerth W, Klie H, Wheeler M, Stoffa P, Sen M (2006) On optimization algorithms for the reservoir oil well placement problem. Comput Geosci (in press)
65. Ji XD, Familoni BD (1999) A diagonal recurrent neural network-based hybrid direct adaptive SPSA control system. IEEE Trans Autom Control 44:1469–1473
66. Maeda Y, Toshiki T (2003) FPGA implementation of a pulse density neural network with learning ability using simultaneous perturbation. IEEE Trans Neural Netw 14:688–695
67. Chunduru RK, Sen MK, Stoffa PL (1997) Hybrid optimization methods for geophysical inversion. Geophysics 62:1196–1207
68. Parashar M, Browne J (2005) Conceptual and implementation models for the grid. Proc IEEE Spec Issue Grid Comput 93(3):653–668
69. Liu H, Parashar M (2006) Accord: a programming framework for autonomic applications. IEEE Trans Syst Man Cybern Spec Issue Eng Auton Syst 36(3):341–352
70. Liu H, Parashar M, Hariri S (2004) A component-based programming framework for autonomic applications. In: the 1st IEEE international conference on autonomic computing (ICAC-04), New York, pp 10–17
71. Liu H, Parashar M (2005) A framework for rule-based autonomic management of parallel scientific applications. In: The 2nd IEEE international conference on autonomic computing (ICAC-05), Seattle, WA, USA
72. Liu H, Parashar M (2005) Rule-based monitoring and steering of distributed scientific applications. Int J High Perform Comput Netw 3(4):272–282
73. Liu H, Parashar M (2005) Enabling self-management of component-based high-performance scientific applications. In: The 14th IEEE international symposium on high performance distributed computing (HPDC-14). Research Triangle Park, NC, pp 59–68
74. Liu H, Bhat V, Parashar M, Klasky S (2005) An autonomic service architecture for self-managing grid applications. In: Proceedings of the 6th IEEE/ACM international workshop on grid computing (Grid 2005). Seattle, WA, USA
75. Chen J, Silver D, Parashar M (2003) Real time feature extraction and tracking in a computational steering environment. In: Proceedings of the 11th high performance computing symposium (HPC 2003), Orlando, FL
76. Liu H, Jiang L, Parashar M, Silver D (2005) Rule-based visualization in the discover computational steering collaboratory. J Future Gener Comput Syst Spec Issue Eng Auton Syst 21(1):53–59
77. Chandra S, Parashar M, Yang J, Zhang Y, Hariri S (2005) Investigating autonomic runtime management strategies for SAMR applications. Int J Parallel Programm 33(2-3):247–259
78. Jiang N, Parashar M (2004) Enabling applications in sensor-based pervasive environments. In: Proceedings of BROADNETS 2004: workshop on broadband advanced sensor networks (BaseNets 2004), San Jose, CA, USA
79. Schmidt C, Parashar M (2004) Enabling flexible queries with guarantees in P2P systems. IEEE Netw Comput Spec Issue Inform Dissem Web 8(3):19–26
80. Mann V, Parashar M (2003) DISCOVER: a computational collaboratory for interactive grid applications. In: Berman F, Fox G, Hey T (eds) Grid computing: making the global infrastructure a reality. Wiley, New York, pp 727–744
81. Parashar M, Muralidhar R, Lee W, Wheeler M, Arnold D, Dongarra J (2005) Enabling interactive oil reservoir simulations on the grid. Concur Comput Pract Exp 17(11):1387–1414
82. Foster I, Kesselman C, Nick J, Tuecke S (2002) Grid services for distributed system integration. IEEE Comput 36(6):37–46
83. Foster I, Kesselman C, Nick JM, Tuecke S (2002) The physiology of the grid: an open grid services architecture for distributed systems integration. http://www.globus.org/research/papers/ogsa.pdf
84. Graham S, Simeonov S, Boubez T, Davis D, Daniels G, Nakamura Y, Neyama R (2002) Building Web services with Java: making sense of XML, SOAP, WSDL, and UDDI. SAMS Publishing, USA
85. Data Access and Integration Services. //http://www.cs.man.ac.uk/grid-db/documents.html
86. 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
87. 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
88. Smith J, Gounaris A, Watson P, Paton NW, Fernandes A, Sakellariou R. Distributed query processing on the grid. http://www.cs.man.ac.uk/grid-db/documents.html

89. Narayanan S, Kurc T, Catalyurek U, Saltz J (2003) Database support for data-driven scientific applications in the grid. Parallel Process Lett 13(2):245–271
90. Weng L, Agrawal G, Catalyurek U, Kurc T, Narayanan S, Saltz J (2004) An approach for automatic data virtualization. In: The 13th IEEE international symposium on high-performance distributed computing (HPDC-13)
91. Parashar M, Matossian V, Bangerth W, Klie H, Rutt B, Kurc T, Catalyurek U, Saltz J, Wheeler M (2005) Towards dynamic data-driven optimization of oil well placement. In: Sunderam V et al (eds) Proceedings of the workshop on distributed data driven applications and systems. International conference on computational science 2005 (ICCS 2005), vol 3514–3516, Springer, Berlin Heidelberg New York, pp 656–663