Cluster Computing 8, 255–269, 2005

An Autonomic Reservoir Framework for the Stochastic Optimization of Well Placement

WOLFGANG BANGERTH and HECTOR KLIE
Center for Subsurface Modeling, The University of Texas at Austin, Austin, TX

VINCENT MATOSSIAN and MANISH PARASHAR
The Applied Software Systems Laboratory, Rutgers University, Piscataway, NJ

MARY F. WHEELER
Center for Subsurface Modeling, The University of Texas at Austin, Austin, TX

Abstract. The adequate location of wells in oil and environmental applications has a significant economic impact on reservoir management. However, the determination of optimal well locations is both challenging and computationally expensive. The overall goal of this research is to use the emerging Grid infrastructure to realize an autonomic self-optimizing reservoir framework. In this paper, we present a policy-driven peer-to-peer Grid middleware substrate to enable the use of the Simultaneous Perturbation Stochastic Approximation (SPSA) optimization algorithm, coupled with the Integrated Parallel Accurate Reservoir Simulator (IPARS) and an economic model to find the optimal solution for the well placement problem.

Keywords: Grid computing, autonomic Grid middleware, stochastic optimization, optimal well placement, reservoir management

1. Introduction

The locations of wells in oil and environmental applications significantly affect the productivity and environmental/economic benefits of a subsurface reservoir. However, the determination of optimal well locations is a challenging problem since it depends on geological and fluid properties as well as on economic parameters. This leads to a very large number of potential scenarios that must be evaluated using numerical reservoir simulations. Reservoir simulators are based on the numerical solution of a complex set of coupled nonlinear partial differential equations over hundreds of thousands to millions of gridblocks. The high costs of simulation make an exhaustive evaluation of all these scenarios infeasible. As a result, the well locations are traditionally determined by analyzing only a few scenarios. However, this *ad hoc* approach may often lead to incorrect decisions with a high economic impact.

Optimization algorithms offer the potential for a systematic exploration of a broader set of scenarios to identify optimum locations under given conditions. These algorithms together with the experienced judgment of specialists, allow a better assessment of uncertainty and significantly reduce the risk in decision-making. Consequently, there is an increasing interest in the use of optimization algorithms for finding the optimum well location in oil industry [4,8,17,32]. However, the selection of appropriate optimization algorithms, the runtime configuration and invocation of these algorithms, and the dynamic optimization of the reservoir remain a challenging problem.

The overall goal of this research is to use the emerging Grid infrastructure [7] and its support for seamless aggregations, compositions and interactions, to realize an autonomic self-optimizing reservoir application. The application consists of: (1) sophisticated reservoir simulation components that encapsulate complex mathematical models of the physical interaction in the subsurface, and execute on distributed computing systems on the Grid; (2) Grid services that provide secure and coordinated access to the resources required by the simulations; (3) distributed data archives that store historical, experimental and observed data; (4) sensors embedded in the instrumented oilfield providing real-time data about the current state of the oil field; (5) external services that provide data relevant to optimization of oil production or of the economic profit such as current weather information or current prices; and (6) the actions of scientists, engineers and other experts, in the field, the laboratory, and in management offices.

These components need to dynamically discover one another and interact as peers to achieve the overall application objectives. First, the simulation components interact with Grid services to dynamically obtain necessary resources, detect current resource state, and negotiate required quality of service. Next, we recall that the data necessary for reservoir simulation is usually sparse and incomplete; in particular, this concerns the data on the geology of the subsurface and on the resident fluids which are very difficult to obtain. Therefore, the simulation components interact with one another and with data archives and real-time sensor data to enable better characterization of the reservoir through processes of dynamic data injection, and data driven adaptations. Then,

the reservoir simulation components interact with other services on the Grid, for example, with optimization services to optimize well placement, with weather services to control production, and with economic modeling services to detect current and predicted future oil prices so as to maximize the revenue from the production. Finally, the experts (scientists, engineers, and managers) collaboratively access, monitor, interact with, and steer the simulations and data at runtime to drive the discovery process.

The overall oil production process described above is autonomic in that the peers involved automatically detect suboptimal oil production behaviors at runtime and orchestrate interactions among themselves to correct this behavior. Further, the detection and optimization process is achieved using policies and constraints that minimize human intervention. The interactions between instances of peer services are opportunistic, based on runtime discovery and specified policies, and are not predefined.

In this paper we use our prototype autonomic reservoir framework [15] to investigate the policy-driven runtime selection and invocation of optimization services to determine optimal well placement and configuration. The specific objectives of this paper include: (1) characterization of the behavior and applicability of optimization techniques for oil reservoir optimization; (2) formulation of policies for the runtime selection and invocation of optimization services for well placement; and (3) the design of a prototype policy-driven framework for autonomic reservoir optimization in Grid environments. In our earlier work [15], we studied the use of the Very Fast Simulated Annealing (VFSA) [24] optimization technique. In this paper we use the Simultaneous Perturbation Stochastic Approximation (SPSA) [25,27] algorithm for optimizing well placement.

The reservoir framework consists of (i) instances of distributed multi-model, multi-block reservoir simulation components provided by the IPARS reservoir simulator framework, (ii) optimization services based on the SPSA algorithm, (iii) economic modeling services, (iv) real-time services providing current economic data (e.g. oil prices), (v) archives of data that has already been computed, and (vi) experts (scientists, engineers) connected via pervasive collaborative portals. It is built on the Pawn P2P substrate, which provides JXTA-based [22] peer-to-peer messaging services, and the Discover computational collaboratory, which combines Grid infrastructure services provided by Globus [6] and interaction and collaboration services.

The rest of this paper is organized as follows. Section 2 describes the well placement problem and introduces the underlying models and components. It also presents the SPSA optimization algorithm. Section 3 describes the design and implementation of the autonomic reservoir framework. Sections 4 describes the well location optimization process using SPSA. Section 5 derives policies for the selection and invocation of optimization services for autonomic well placement. Section 6 presents a summary and conclusions.

2. Autonomic oil well placement optimization

In this section, we specify the mathematical models underlying the reservoir simulation (forward model), the revenue function (objective function), and the stochastic optimization algorithm. We end the section with a description of the case study based on a real application problem.

2.1. Problem description

Let us assume that there exists an oil reservoir whose properties are known, at least at a given scale, and in which a few wells are already operating. The problem is to find the optimum geographical location for drilling a new well in order to maximize production, oil sweep efficiency or a given revenue value. In practice, the question of finding optimal operating schedules of new and existing wells, i.e. for example pumping rates as a function of future time, is also important, but is a much more complicated problem that we will not consider here. We will also only look at the placement of one well at a time.

The well placement problem is an optimization problem for the well location $p = (x, y)$, which has to lie in a set P of possible parameter values. In order to describe what we mean by "optimal well location", we need to define a scalar objective function $f(p)$ that measures the economic cost of drilling and operating at position p minus the revenue we get from the produced oil. The goal is then to minimize this function, or equivalently to maximize the revenue minus the cost. We will describe this objective function in Section 2.3.

With this function defined, the optimization problem consists of finding that position $p_{\text{opt}} \in P$ such that the cost $f(p_{\text{opt}})$ is less than or equal to the cost $f(p)$ for all other possible source locations $p \in P$. The task of finding this optimum is complicated by three facts:

- First, the set P does not necessarily have to be continuous; rather, it can, and in fact it will in the example shown below, consist of single points because our numerical model only allows us to place wells at a discrete set of positions (the only viable locations are the centers of cells of our finite element scheme). This discreteness of the set P of course precludes the computation of derivatives.
- Secondly, even if P is a continuous set, derivatives of $f(p)$ are usually unavailable analytically because of the complexity of computing them; in addition, $f(p)$ may not be differentiable at all, rendering the question of computing derivatives moot. Therefore we focus on a class of gradient-free optimization methods which require only the evaluation of the objective function $f(p)$ at certain points. This task is accomplished by running a reservoir simulator for a number of trial positions p and evaluating the economic objective function $f(p)$ for the predicted production of a model with a well at position p.
- Thirdly, computing function values for models as the ones considered here is expensive: for realistic simulations,

evaluating the objective function for a given well location can easily take many hours even on fast computers. This forces us to make use of efficient optimization methods, as well as novel approaches to distributed computing. In the model application considered here, we use a simplified model that reduces the computing time for one evaluation of $f(p)$ to about 25 minutes on an AMD Athlon 2 GHz Linux-based desktop computer. This reduction in complexity enables us to completely map the objective function for all possible well locations in order to verify the path the optimizer is describing. However, this is neither possible nor economic in realistic applications and it is only used in this paper to illustrate the effectiveness of the method.

In the following, we provide a brief overview of the mathematical models and optimization methods. We note that these two parts are essentially independent of one another: the simulator just computes $f(p)$ for a given $p \in P$, without knowledge of what will be done with this value; on the other hand, the optimizer just asks for $f(p)$ for a given p, without caring how it is computed. This independence is reflected in the implementation by making the reservoir simulation model and the optimizer two independent components that interact only by using the Pawn interaction middleware.

2.2. Mathematical model for the flow in an oil reservoir

We consider a heterogeneous 3D oil reservoir, denoted by Ω, surrounded by impermeable rocks (i.e., no flow boundary conditions). The set of partial differential equations describing the conservation of mass of each component $m = o, w$ (oil and water) are

$$\frac{\partial(\phi N_m)}{\partial t} + \nabla \cdot U_m = q_m. \tag{1}$$

Here, ϕ is the porosity of the porous medium, N_m the concentration of a component m, and q_m the sources (production and injection rates). The fluxes U_m are defined using Darcy's law [9] which, with gravity ignored, reads as $U_m = -\rho_m K \lambda_m \nabla P_m$, where ρ_m denotes the density of a component, K the permeability tensor, λ_m the mobility of a component, and P_m the pressure of a phase. Additional equations specifying volume, capillary, and state constraints are added, and boundary and initial conditions complement the system, see [2,9]. Finally, $N_m = S_m \rho_m$ with S_m denoting saturation of a phase. The resulting system (omitting gravity terms for simplicity) is

$$\frac{\partial(\phi \rho_m S_m)}{\partial t} - \nabla \cdot (\rho_m K \lambda_m \nabla P_m) = q_m. \tag{2}$$

In this paper we consider wells that either produce (a mixture of) oil and water, or at which water is injected. At an injection well, the source term q_w is nonnegative (we will use the notation $q_w^+ := q_w$ to make this explicit). At a production well, both q_o and q_w may be non-positive and we will denote this by $q_m^- := -q_m$. In practice, both injection and production rates are subject to control, and thus to optimization; however, in this paper we assume that rates are user predefined and are not decision parameters in our problem.

This model is discretized in space using the *expanded mixed finite element* method which, in the case considered in this paper, is numerically equivalent to the *cell-centered finite difference* approach [1,23]. Time discretization can be either fully implicit, semi-implicit or sequential; here we only consider the sequential method in which two linear systems of equations, the pressure equation and the concentration equation, are solved at each time step.

This discrete model is solved by the IPARS (Integrated Parallel Accurate Reservoir Simulator) software developed at the Center for Subsurface Modeling at The University of Texas at Austin [10,13,19,21,28–31]. IPARS is a parallel reservoir simulation framework for modeling multiphase, multiphysics flow in porous media. It offers sophisticated simulation components that encapsulate complex mathematical models of the physical interaction in the subsurface, and which execute on parallel and distributed systems. Solvers employ state-of-the-art techniques for nonlinear and linear problems including multigrid and other preconditioners [11]. It can handle an arbitrary number of wells each with one or more completion intervals. Although not used here, IPARS supports multiple physical models and their multiphysics couplings.

2.3. The economic model

In general, the economic value of production is a function of the time of production and of injection and production rates in the reservoir. It takes into account fixed costs such as drilling a well, prices of oil, costs of injection, extraction, and disposal of water, as well as associated operating costs. We assume here that operation and drilling costs are fixed, i.e. independent of the well location.

We therefore define our objective function by summing the revenues from produced oil over all production wells, and subtracting the costs of disposing produced water and the cost of injecting water. We then obtain

$$f(p) = -\int_0^T \left\{ \sum_{\text{prod. wells}} \{(c_o q_o^-(s) - c_{w,\text{disp}}\, q_w^-(s))\} - \sum_{\text{inj. wells}} c_{w,\text{inj}}\, q_w^+(s) \right\} (1+r)^{-t} dt, \tag{3}$$

where q_o^- and q_w^- are production rates for oil and water, respectively, and q_w^+ are injection rates, each in barrel per day. The coefficients $c_o = 24$, $c_{w,\text{disp}} = 1.5$ and $c_{w,\text{inj}} = 2$ are the prices of oil and the costs of disposing and injecting water, in dollars per barrel each. The exponential factor takes into account that the drilling costs have to be paid up front and have to be paid off with interest. We choose an interest rate of $r = 10\% = 0.1$ per year. T is the time horizon up to which we perform our simulations, and up to which we integrate the revenue. Finally, we define $f(p)$ to be the negative total

revenue, since we want to minimize $f(p)$, which then amounts to maximizing the revenue.

Note that $f(p)$ depends on the location p of the additional well in two ways. First, the injection rates of the additional well, and thus its associated costs, depend on its location if the bottom hole pressure (BHP) is prescribed. Secondly, the production rates of the other wells as well as their water-oil ratio depend on where water is injected.

We remark that other objective functions would also be possible. For example, one may want to minimize the amount of bypassed oil, i.e. oil that is not going to be produced from the reservoir by the given set of wells. Or, one may wish to minimize the amount of produced water. This last case is somewhat akin to preventing the *water coning* and *water fingering* phenomena [5, 20]. Note, however, that the (negative) cost of water production already appears as one term in the objective function defined above.

2.4. Optimization

As mentioned above, viable methods for finding the maximum or minimum of our objective function $f(p),\ p \in P$, must be content with evaluating $f(\cdot)$ directly since gradients are not available. In addition, we are only interested in methods that are *efficient*, i.e. need only a small number of function evaluations, in order to keep computing times within a manageable range. In a previous study [15], we have used the Very Fast Simulated Annealing (VFSA) algorithm to find the minimum of $f(p)$. Here, we focus on the use of the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm, see [25, 27].

Stochastic approximation (SA) methods represent an important class of stochastic search algorithms. Many well-known techniques are special cases of SA, including neural-network backpropagation, perturbation analysis for discrete-event systems, recursive least squares and least mean squares, genetic algorithms and simulated annealing. SPSA works by starting from an initial guess $p_0 \in P$ and then in each iteration k performing the following steps:

Algorithm 2.1 (SPSA)**.**

1 Set $k = 1, \gamma = 0.101, \alpha = 0.602$.

2 While $k < K_{\max}$ or convergence has not been reached do

- **2.1** Compute a random search direction Δ_k in $\{-1, +1\}$.
- **2.2** Compute $c_k = \frac{c}{k^\gamma}, a_k = \frac{a}{k^\alpha}$.
- **2.3** Evaluate $f^+ = f(p_k + c_k\Delta_k)$ and $f^- = f(p_k - c_k\Delta_k)$.
- **2.4** Compute an approximation to the magnitude of the gradient by $g_k = (f^+ - f^-)/2c_k$.
- **2.5** Set $p_{k+1} = p_k - a_k g_k \Delta_k$.
- **2.6** Set $k = k + 1$.

end while

Some comments are in order. Step 2.1 selects each vector component of Δ_k to be independent and satisfy certain statistical properties. The simplest choice that satisfies these requirements is to choose them from a Bernoulli distribution, i.e., Δ_k in $\{-1, +1\}$. The gain parameters c_k, a_k are decreasing sequences with respect to k. Although they may change according to the problem, we have found it suitable to define them as suggested in [26]. For the present problem, we use $c = 5$ and $a = 2 \cdot 10^{-5}$. Step 2.3 and 2.4 are used to compute an approximation g_k to the magnitude of the gradient. The reader may realize that the update of the solution in Step 2.5 is basically a stochastic version of a steepest descent method (see [27]).

In other words, in each step the algorithm chooses a random direction and looks ahead and back a certain distance c_k in this direction for the value of the objective function $f(\cdot)$. Depending on whether the function value is smaller in the forward or backward direction, it moves the next iteration forward or backward by $a_k g_k$. In practice, we stop the iteration if it did not make any significant progress in the last κ steps (i.e. cycles back and forth), measured by the criterion $|p_k - p_{k-\kappa}| < \xi$; in our computations, we chose $\kappa = 6$ and $\xi = 2$. Note that we do not necessarily stop at an optimum but rather at some random point while jumping back and forth; however, both the stopping point as well as the best point encountered during the process are usually very close in value to the global optimum.

The success of this algorithm is due to the fact that even though it only uses two function evaluations per iteration and uses random directions, it always generates a descent direction (at least with respect to the given step length). It is thus able to approximate the gradient of $f(\cdot)$ without actually computing it, by generating random directions that, on average, resemble the gradient.

As mentioned above, we only consider a discrete and finite set P for the possible well locations. Thus, the above algorithm requires two modifications:

- c_k and $a_k g_k$ need to be integers. To enforce this, we always round these values up to the next integer, i.e. we use $\lceil \frac{c}{k^\gamma} \rceil$, $\lceil \frac{a}{k^\alpha} g_k \rceil$ where c_k and $a_k g_k$ appear. This, together with the choice of Δ_k makes sure that all iterates and evaluation points are on the integer lattice on which we optimize.
- Iterates and evaluation points have to stay within the bounds surrounding P. For this, let $\Pi(p)$ be the closest point in P for a given point p (which may lie outside of P). Then we use $f^+ = f(\Pi(p_k + c_k\Delta_k))$ and $f^- = f(\Pi(p_k - c_k\Delta_k))$. The new step is computed as $p_{k+1} = \Pi(p_k \pm a_k g_k \Delta_k)$. Since our feasible region P is the set of integers inside a box, this simple procedure always guarantees that we find a viable step.

With these modifications, the algorithm only ever evaluates points that are members of the set P.

We note that in the present context of distributed peer-to-peer applications, SPSA has a number of advantages compared to some other optimization algorithms, for example the VFSA algorithm mentioned above [15]. In particular, in Step 2.4 of the SPSA algorithm outlined above, we need to perform two function evaluations, each of which requires running IPARS for a given well location. Since these computations are

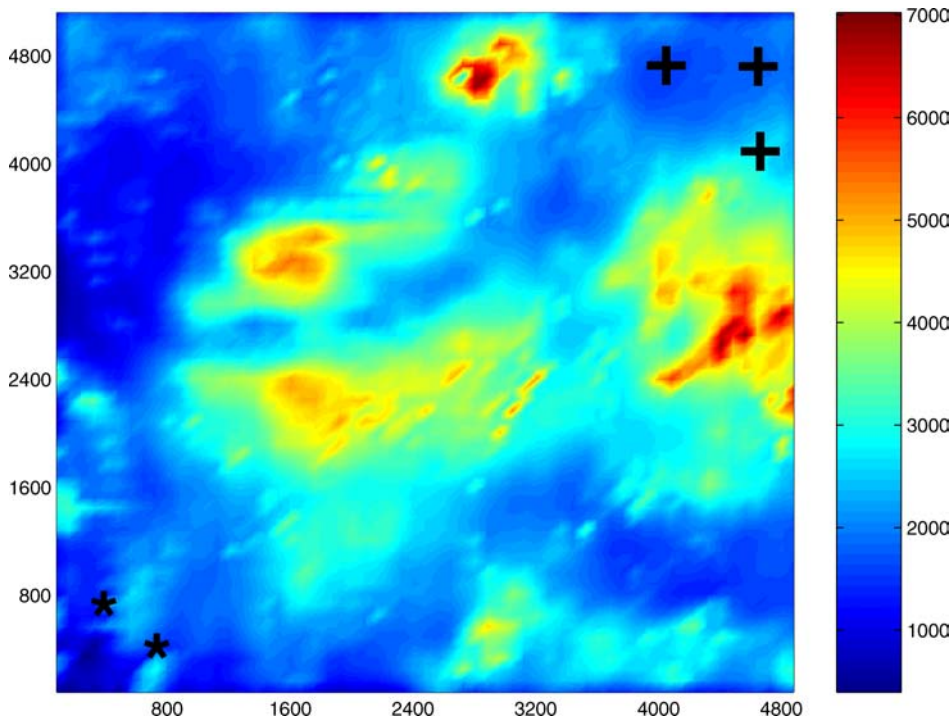

Figure 1. Permeability field showing the positions of current wells. The symbols '$*$' and '$+$' indicate injection and producer wells, respectively.

independent, they could well be run in parallel, for example on two different clusters. Given the high cost of running each of these simulations, this can reduce the run-time by a factor of two. Also, there are modifications of the basic SPSA algorithm that not only compute *one* search direction Δ_k and evaluate the objective function in forward and backward direction, but rather generate *several*, say S search directions, resulting in $2S$ function evaluations [25]. The final update step from p_k to p_{k+1} is then done by incorporating the information of all these computations. This modification allows a better approximation of the true gradient of $f(\cdot)$ and will thus converge in less iterations. The cost of additional function evaluations could be buffered by running some or all of the independent $2S$ IPARS computations in parallel, a task which the IPARS Factory (to be described below) could easily distribute to available resources. Finally, by starting at different initial points, the algorithm may converge to the same optimum solution (augmenting the reliability of reaching a unique global solution) or to a set of different solutions (several extrema due to the nonlinearity of the problem). In the latter case, specialists and management could be interested in looking at clusters of solutions for comparison against other complex factors not included during the optimization stage. We have not yet implemented these extensions to the basic SPSA algorithm, but plan to explore them in a future work.

2.5. Case study

In our case study we consider a 2D reservoir $\Omega = [0, 4880] \times [0, 5120]$ of roughly 25 million ft^2, which is discretized by a 61×64 spatial grid of 80 ft spacing along each horizontal direction, and a depth of 30 ft. Hence, the model consists of 3904 gridblocks. The reservoir under study is located at a depth of 3868.94 ft (i.e., 1 km) and corresponds to a 2D section extracted from the Gulf of Mexico. The porosity has been fixed at $\phi = 0.2$ but the reservoir has a heterogeneous permeability field as shown in figure 1. The fluids are initially in equilibrium with water pressures set to 2600 psi and oil saturation to 0.7.

The original reservoir consists of 5 wells: 2 water injectors and 3 oil producers. Figure 1 shows the opposite-corner distribution of injectors (bottom left) and producers (top right). Injection and production rates are computed by specifying a fixed bottom hole pressure (BHP). Since oil flows from the lower left corner to the upper right corner, one would intuitively guess that the new injection well should be located somewhere in the neighborhood of the reservoir center. The permeability field suggests that flow should be faster in the lower part of the reservoir, so the new well should shift its location to the upper part, where oil is displaced more slowly. This is also indicated by looking at the oil saturation and pressures at the end of the simulation period at $T = 2000$ days, as shown in figure 2. However, such analysis is not that straightforward when more wells are involved.

Given this description of the domain, the parameter space is the set of 3904 points of the integer lattice $P = \{40, 80, 120, \ldots, 4840\} \times \{40, 80, 120, \ldots, 5080\}$ of cell midpoints, at which we can place wells in our computational model. We assume that the well penetrates through the entire depth of the reservoir, that is, the depths of its bottom and top are fixed. We also fix the BHP operating conditions at the

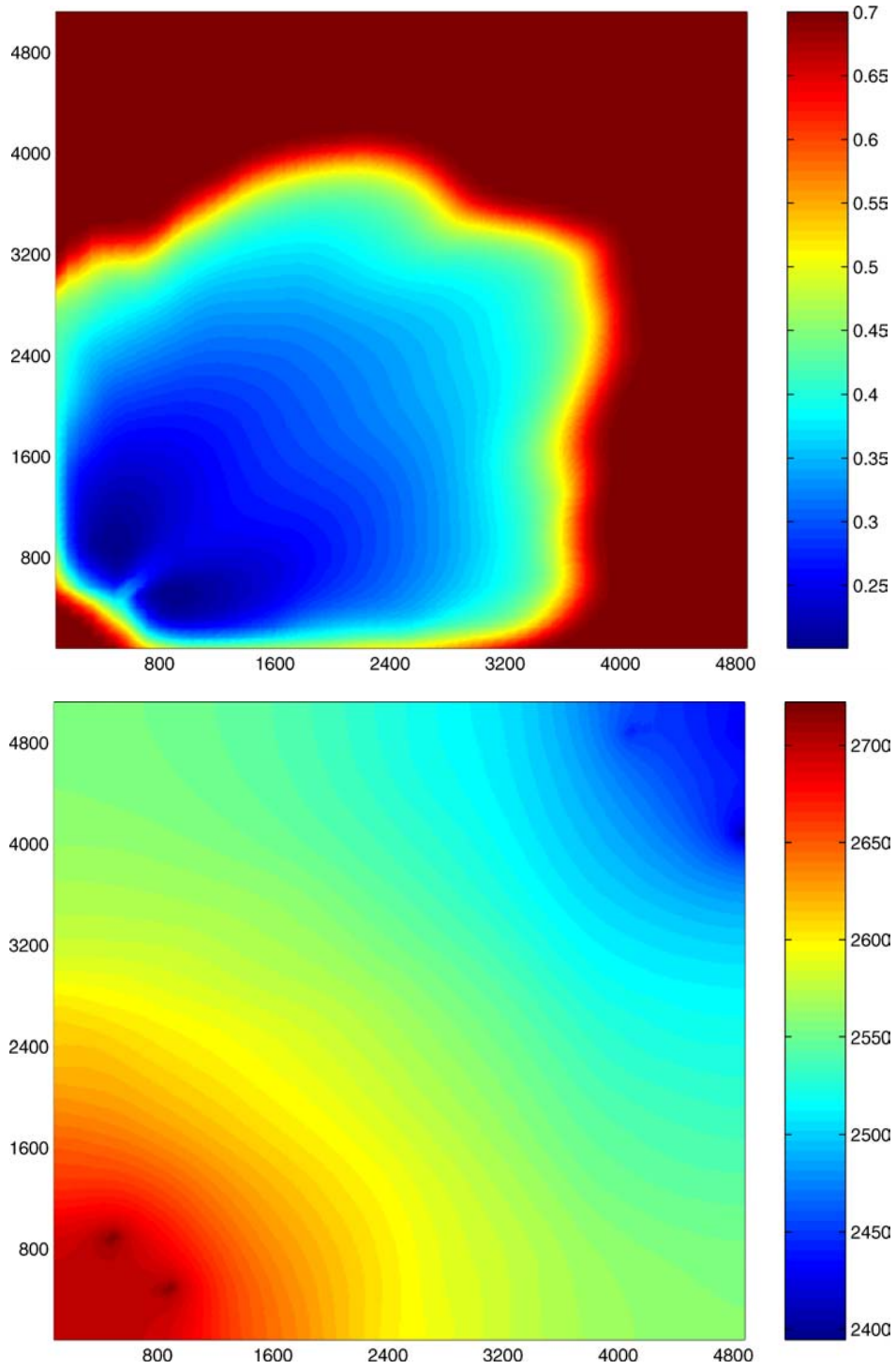

Figure 2. Top: Oil saturation at the end of the simulation for the original well distribution. Bottom: Oil pressure.

new injection well to be the same as that at the other injection wells. We note that in general, the BHP and well penetration parameter could vary and become an element of P. Also, more wells could be placed.

The goal of the case study is then to find the optimal position $p \in P$ of a new well, with respect to the objective function $f(p)$ defined above. Given enough computing resources, one could evaluate $f(p)$ for all 3904 possible $p \in P$ and from this easily determine the optimal well location. For the simple test case considered here where every function evaluation takes about 25 min on a Linux PC consisting of dual 2 GHz AMD Athlon processors, we have actually done this and show the results in figure 3. However, for more realistic computations, this is of course not possible, and optimization algorithms

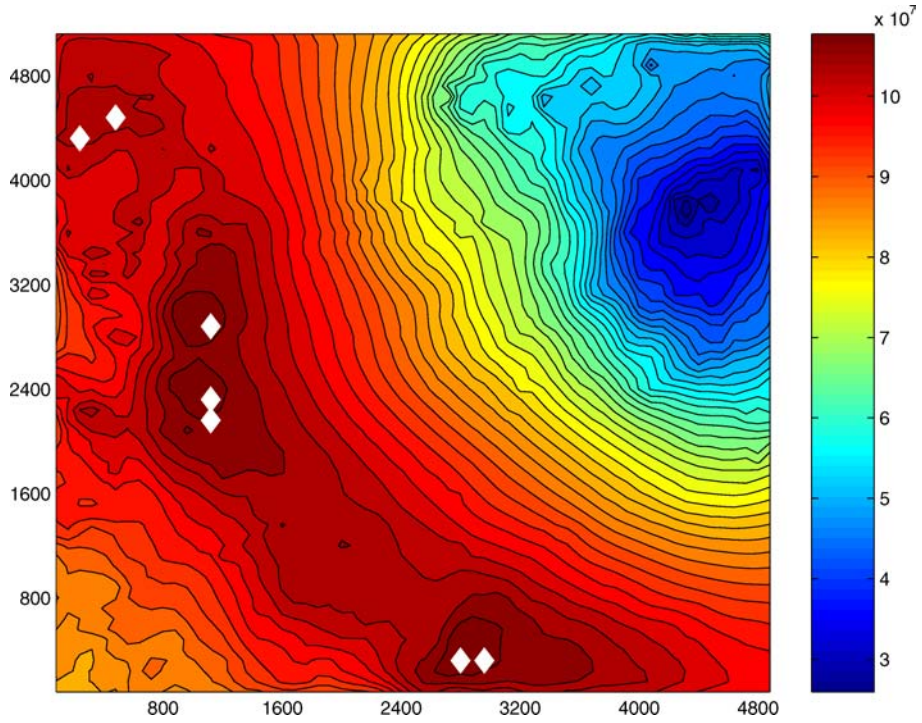

Figure 3. Search space response surface: Expected revenue $-f(p)$ for all possible well locations $p \in P$. White marks indicate optimal well locations found by SPSA for 7 different starting points of the algorithm.

have to use much less than this number of function evaluations. In this paper we achieve this using the SPSA algorithm discussed above.

Note that while we would in general like to compute the global optimum, we will usually be content if the algorithm finds a solution that is almost as good. This is important in the present context where the revenue surface plotted in figure 3 has 72 local optima, with the global optimum being $f(p = \{2920, 920\}) = -1.09804 \cdot 10^8$. However, there are 5 more local extrema within only half a per cent of this optimal value, which makes finding the global optimum rather complicated. The white marks in the figure indicate the best well positions found by the SPSA algorithm when started from seven different points on the top-left to bottom-right diagonal of the domain. As can be seen, SPSA is able to find very good well locations from arbitrary starting points, even though it does not find the global optimum every time.

3. Enabling autonomic oil reservoir optimization using decentralized services

The overall application scenario is illustrated in figure 4. The primary peers and services participating in the application are described below.

3.1. Integrated Parallel Accurate Reservoir Simulator (IPARS)

IPARS is the reservoir simulator that, together with the economic model, is used to evaluate the objective function. It is a peer in our application that takes a number of input files which, among other things, specify a well position p, and returns the production history of all wells. IPARS is primarily implemented in Fortran and C, but is integrated with the framework discussed in this paper using C++ wrappers and the Java Native Interface.

3.2. IPARS factory

The IPARS Factory is responsible for configuring instances of IPARS simulations, deploying them on resources on the Grid, and managing their execution. Configuration consists of generating the relevant input files that select appropriate models from those provided by IPARS, define the structure and properties of the reservoir to be simulated, and list required parameters. Deployment and management of IPARS instances use services provided by Discover [14] and Globus [6], and build on the CORBACoG Kit [18].

3.3. SPSA optimization service

The SPSA Optimization service runs on the Optimization peer and implements the SPSA algorithm presented in Section 2.4. It also offers interfaces and mechanisms for interactive and autonomic communications between the Optimization peer, IPARS instances, and the IPARS Factory. The optimization service uses the SPSA algorithm to generate guesses of new well positions. This guess is first compared against an archive of already computed well positions, therefore preventing useless computation of already known data. If no match is found,

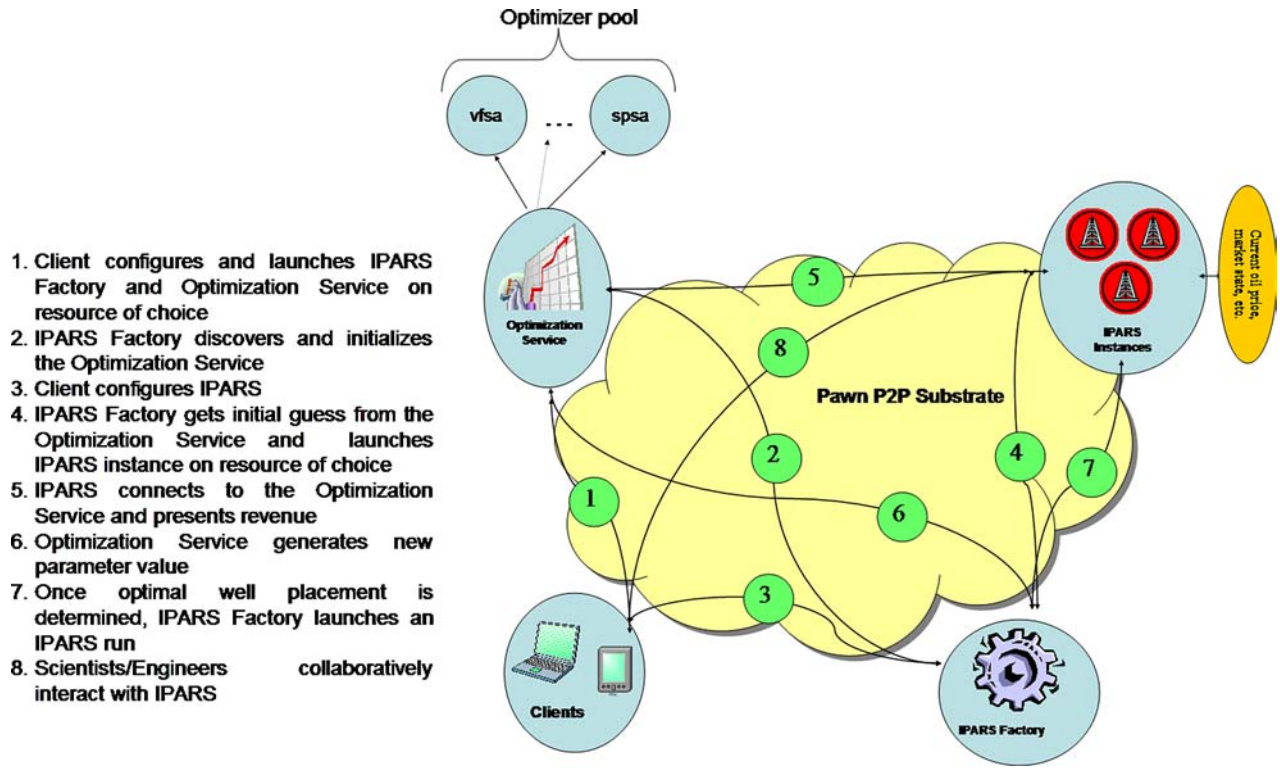

Figure 4. Autonomous oil reservoir optimization using decentralized services.

the new guess is added to the archive and is forwarded to the IPARS Factory. The IPARS factory then uses these well positions to initialize and configure a new instance of IPARS.

3.4. Economic modeling service

The Economic Modeling Service is based on the economic model presented in Section 2.3 and uses the output produced by an IPARS simulation instance and current market parameters (e.g. oil prices, drilling costs, etc.) to compute estimated revenues for a particular reservoir configuration.

The market parameters used by the model are variable economic indices including the price of oil per volume produced, the cost of water per volume, the cost of disposal of water, and the current discount rate. These indices are obtained using a network information service that collects information at regular intervals from different sources on the Internet. The network information service is implemented as a threaded Java Servlet and is part of the Discover middleware. The Servlet essentially queries a relevant URL (e.g. http://money.cnn.com/markets/commodities.html), and parses the responses to extract current oil, gas and water prices. This information is then fed into the economic model during the optimization process.

In general, instead of fixed current prices obtained by the network information services, one may be able to use a set of "forecasts" of prices, delivered by stochastic or other mathematical models. This would allow a more realistic planning of future revenues from an oil field. However, this capability is not currently implemented and is not a part of the prototype application.

3.5. Discover computational collaboratory

Discover [14] is a virtual, interactive computational collaboratory that provides services to enable geographically distributed scientists and engineers to collaboratively monitor and control high performance parallel/distributed applications on the Grid. Its primary goal is to bring Grid applications to the scientists'/engineers' desktops, enabling them to collaboratively access, interrogate, interact with, and steer these applications using pervasive portals. Key components of the Discover collaboratory include:

- *Discover Interaction & Collaboration Middleware Substrate* [3] that enables global collaborative access to multiple, geographically distributed instances of the Discover computational collaboratory, and provides interoperability between Discover and external Grid services. The middleware substrate enables Discover interaction and collaboration servers to dynamically discover and connect to one another to form a peer network. This allows clients connected to their local servers to have global access to all applications and services across all servers based on their credentials, capabilities and privileges.

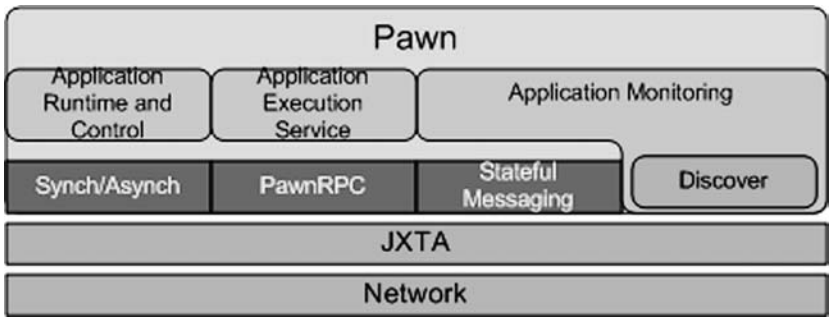

Figure 5. Pawn architecture: Pawn builds on network and interaction services to enable P2P interactions in Grid applications.

The Discover middleware also integrates Discover collaboratory services with the Grid services provided by the Globus Toolkit [6] using the CORBA Commodity Grid (CORBA CoG) Kit [18]. Clients can use the services provided by the CORBA CoG Kit to discover available resources on the Grid, to allocate required resources, to run applications on these resources, and use Discover to connect to and collaboratively monitor, interact with, and steer the applications.

- *DIOS Interactive Object Framework* (*DIOS*) [12,16] that enables the runtime monitoring, interaction and computational steering of parallel and distributed applications on the Grid. DIOS enables application objects to be enhanced with sensors and actuators so that they can be interrogated and controlled. Application objects may be distributed (spanning many processors) and dynamic (be created, deleted, changed or migrated at runtime). A control network connects and manages the distributed sensors and actuators, and enables their external discovery, interrogation, monitoring and manipulation. The control network enables sensors and actuators to be encapsulated within, and directly deployed with the computational objects. The DIOS distributed rule engine allows users to remotely define and deploy rules and policies at runtime and enables autonomic monitoring and steering of Grid applications.
- *Discover Collaborative Portals* [14] that provide the experts (scientists, engineers) with collaborative access to other peer components. Using these portals, experts can discover and allocate resources, configure and launch peers, and monitor, interact with, and steer peer execution. The portal provides a replicated shared workspace architecture and integrates collaboration tools such as chat and whiteboard. It also integrates "Collaboration Streams," that maintain a navigable record of all client-client and client-applications interactions and collaboration.

3.6. Pawn peer-to-peer messaging framework

Pawn builds on Project JXTA [22] and enables peers to exchange messages through common services and interaction modes. Figure 5 shows the services and interaction modalities enabled by the Pawn framework.

Pawn offers four key services to enable dynamic collaborations and autonomic interactions in scientific computing environments.

The *Application Runtime and Control* [*ARC*] announces the existence of an application to the peergroup, sends application responses, publishes application update messages, and notifies the peergroup of an application termination.

The *Application Monitoring and Steering Service* [*AMS*] enables users to interact with an application in real-time. Using the AMS service a user can monitor, retrieve, or set application data.

The *Application Execution Service* [*AEX*] enables a peer to remotely start, stop, get the status of, or restart an application. This service requires a mechanism that supports synchronous and guaranteed remote calls necessary for resource allocation and application deployment (i.e. transaction oriented interactions) in a P2P environment.

The *Collaboration Service* [*Group Communication, Presence*] extends the Discover substrate to provide collaborative tools and support for group communication and detection of presence.

Every peer can implement all or a subset of these services. Particular services subsets characterize a role for the peer. There are three distinct roles that a peer can take:

Client Peer that can deploy applications on available resources for monitoring and/or steering; the client can also collaborate with other peers in the group using Chat and Whiteboard tools.

Application Peer that exports the application interfaces and controls to the peergroup; these interfaces are used by other peers to interact with the application. An application may already be enabled to communicate remotely with a middleware server as in the Discover computational collaboratory [14]; in such a case, the application peer acts as a proxy peer, relaying queries and responses to and from clients to applications.

Rendezvous Peer to distribute or relay messages. Rendezvous peers filter messages as defined by filtering rules input from the connected clients. The communication uses TCP unicast messages between endpoints to establish one-to-one and one-to-many delivery modes.

Using the Pawn and Discover computational collaboratory, clients can connect to a local server using the portal, and can use it to discover and access active applications and services on the Grid as long as they have appropriate privileges and capabilities. Furthermore, they can form or join collaboration groups and can securely, consistently, and collaboratively interact with and steer applications based on their privileges and capabilities. The components described above need to dynamically discover and interact with one another as peers to achieve the overall application objectives. As can be seen in figure 4, the experts use the portals to interact with the Discover middleware and the Globus Grid services to discover and allocate appropriate resource, and to deploy the IPARS Factory, SPSA and Economic Model peers (Step 1). The IPARS Factory discovers and interacts with the SPSA service peer to configure and initialize it (Step 2). The expert interacts with the IPARS Factory and SPSA to define application configuration parameters (Step 3). The IPARS Factory then interacts with the Discover middleware to discover and allocate resources and to configure and execute IPARS simulations (Step 4). The IPARS simulation now interacts with the Economic Model to determine current revenues, and discovers and interacts with the SPSA service when it needs optimization (Step 5). SPSA provides the IPARS Factory with a new guess for a better well location (Step 6), which then uses it to configure and launch new IPARS simulations (Step 7). Experts can, at anytime, discover, collaboratively monitor, and interactively steer IPARS simulations, configure the other services, and drive the scientific discovery process (Step 8). Once the optimal well parameters are determined, the IPARS Factory configures and deploys a production IPARS run.

These interactions are enabled by the Pawn services that build on JXTA's pipe and resolver services to provide *stateful* and *guaranteed* messaging. In Pawn, messages are platform-independent, and are composed of source and destination identifiers, a message type, a message identifier, a payload, and a handler tag. State is maintained by making every message a self-sufficient and self-describing entity that carries enough information such that, in case of a link failure, it can be resent to its destination by an intermediary peer without the need to be recomposed by its original sender. In addition, messages can include system and application parameters in the payload to maintain application state.

Pawn implements application-level communication guarantees by combining stateful messages, message queueing, and a per-message acknowledgment table maintained at every peer. This messaging is used to enable the key application-level interactions such as :

Synchronous/Asynchronous Communication: Communication in JXTA can be synchronous (using blocking pipes) or asynchronous (using non-blocking pipes or the resolver service). In order to provide reliable messaging, Pawn combines these communication modalities with stateful messaging and guarantee mechanism.

Dynamic Data Injection: Pawn leverages JXTA pipes mechanisms and combines it with its guaranteed message delivery mechanism to provide Dynamic Data Injection.

Remote Procedure Calls (*PawnRPC*): The PawnRPC mechanism provides the low-level constructs for building applications interactions across distributed peers. Using PawnRPC, a peer can dynamically invoke a method on a remote peer by passing its request as an XML message through a pipe.

4. Reservoir optimization using the pawn framework

In this section, we describe how Pawn is used to support the prototype autonomic oil reservoir optimization application outlined in Section 2. Every interacting component is a peer that implements Pawn services. The IPARS Factory, SPSA, and the Discover collaboratory are Application peers and implement ARC and AEX services. The Discover portals are Client peers and implement AMS and Group communication services. Key operations in the process include peer deployment (e.g. IPARS Factory deploys IPARS), peer discovery (e.g IPARS Factory discovers SPSA), peer initialization and configuration (e.g. Expert configures SPSA), autonomic optimization (e.g IPARS and SPSA interactively optimize revenue), interactive monitoring and steering (e.g. Experts connect to, monitor, and steer IPARS), and collaboration (e.g. Experts collaborate with one another). These operations are described below.

4.1. IPARS factory and SPSA optimization service deployment

The IPARS Factory and SPSA Optimization peers are deployed using Globus services accessed through Discover/CORBACoG. The SPSA peer is a C++ program that is integrated with Pawn using the Java Native Interface. Figure 6 presents the sequence of operations involved. The deployment is orchestrated by the Expert through the Discover portal. The portal gives the Expert secure access to all the machines registered with Globus Meta Directory Service (MDS) to which the Expert has access privileges. Authentication and authorization is based on the Globus Grid Security Infrastructure (GSI) service. Once authenticated, the Expert can use the portal to deploy the IPARS Factory and SPSA peers on machines of choice after verifying their availability and current status (load, CPU, memory). Deployment uses the Globus GRAM service. The portal also gives the Expert access to already deployed services and applications for collaborative monitoring and steering using Discover.

4.2. Peer initialization and discovery

At startup, peers use the underlying JXTA discovery service to publish an advertisement to the peergroup. This advertisement

Figure 6. Peer deployment.

describes the functionalities and services offered by the peer. It also contains a pipe advertisement for input and output communications, and the RPC interfaces offered by the peer for remote monitoring, steering, service invocation and management. To enable peers to mutually identify each other, the peer that discovers an advertisement sends its advertisement back to the discovered peer. This discovery process is also used by IPARS instances to discover the SPSA service.

4.3. IPARS and SPSA configuration

The Expert uses the portal and the control interfaces exported to configure the SPSA service and to define its operating parameters. The Expert also configures the IPARS Factory by specifying the parameters for IPARS simulations. The IPARS Factory uses these parameters to set up IPARS instances during the optimization process, and initialize the SPSA service. Note that the Expert can always use the interaction and control interfaces to modify these configurations. The configuration uses AMS to send application parameters to the IPARS Factory and SPSA peer. A response is generated and sent back (using AEX) to the client to confirm the configuration change.

4.4. Oil reservoir optimization

The reservoir optimization process consists of two phases, an initialization phase and an iterative optimization phase as described below.

Initialization phase: In the initialization phase, SPSA provides the IPARS Factory with an initial guess of well parameters based on its configuration by the Expert and the IPARS Factory. This is done using the channel established during discovery and is used by the IPARS Factory to initialize and deploy an IPARS instance.

Iterative optimization phase: In the iterative optimization phase, the IPARS instance uses the Economic Model along with current market parameters to estimate the current revenue $f(p)$ for the trial well locations p. SPSA uses this value to generate an updated guess of the well parameters p_{k+1}. It then sends new trial well locations to the IPARS Factory. The IPARS Factory now configures a new instance of IPARS with the updated well parameters and deploys it. This process continues until the required terminating condition is reached. Figure 7 shows the overall optimization process between IPARS Factory, IPARS, and SPSA. Note that experts can connect to any of these peers at any time and steer the optimization process.

Well parameter and revenue archive: After each evaluation of a trial well location, these well parameters and the corresponding revenue computed by IPARS and the Economic Model are stored in an archive (a MySQL database) maintained by an archival peer. During the optimization process, when a new trial location is received from SPSA, the IPARS Factory checks the archive before launching an IPARS instance. If the current location is already present in the archive, the corresponding normalized revenue value is sent back to SPSA and a redundant IPARS instance is avoided.

Note that peer interactions during the optimization process are highly dynamic and require synchronous or asynchronous RPC semantics with guarantees, rather than document exchanges typically supported by P2P systems. In Pawn, these interactions are enabled by PawnRPC, which provides the same semantics as the traditional RPC in a

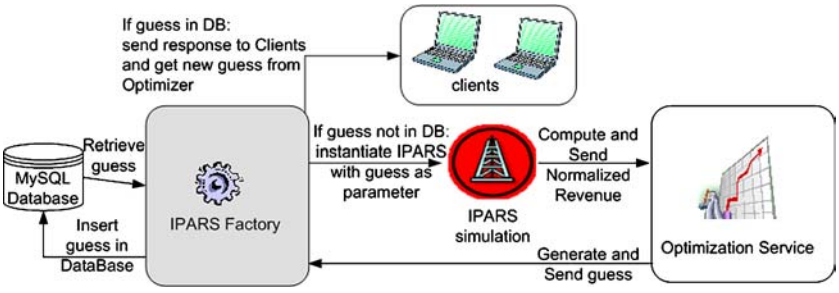

Figure 7. Optimization process.

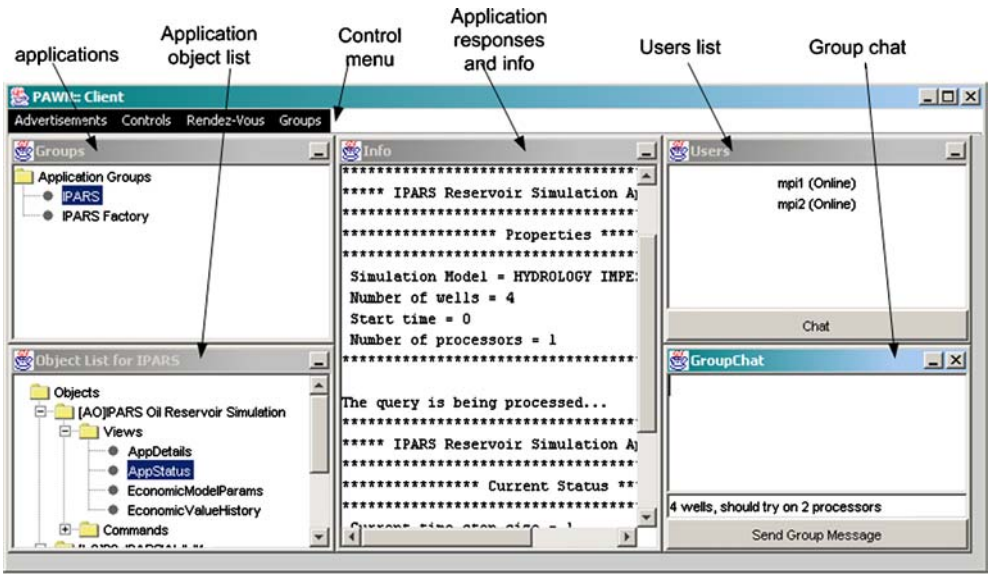

Figure 8. Graphical user interface of the Expert's portal.

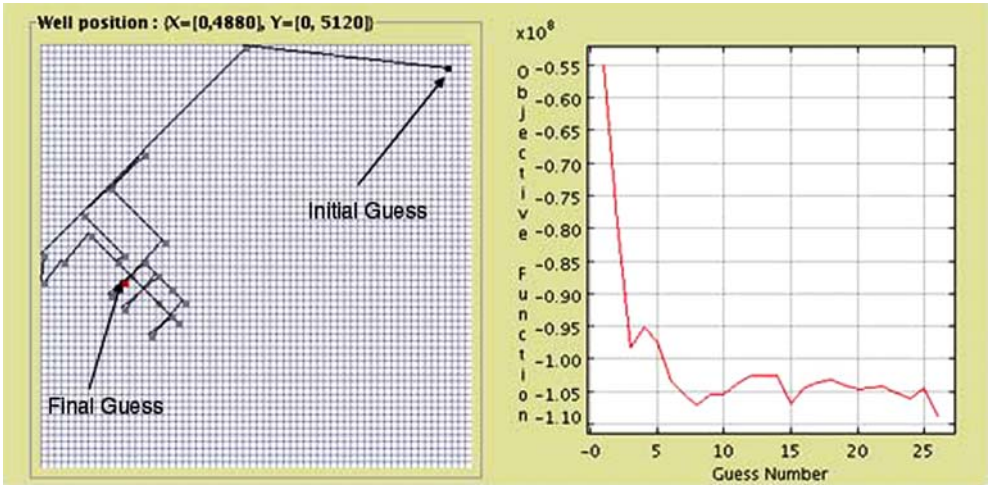

Figure 9. Computed well positions and economic revenue during the optimization process.

client-server system, but is implemented in a purely P2P manner.

4.5. Production runs and collaborative monitoring and steering

Once the optimization process terminates and the optimal well parameters are determined, the IPARS Factory allocates appropriate resources, configures a production run based on these parameter, and launches this run on the allocated resources.

Experts can now collaboratively connect to the running application, collectively monitor its execution and interactively steer it. Figure 8 presents the client peer's portal interface used by the Experts. The portal interface can also be used to access, monitor and steer the IPARS Factory, the SPSA Optimization service, and the Economic Model.

4.6. Sample results from the oil reservoir optimization process

Sample results from the oil reservoir optimization process are shown in figures 3 and 9. The first shows the computed revenue for each possible well location, and the points which SPSA chooses as optimal well locations for a number of different initial guesses. Figure 9 shows the path the SPSA iterates take for a particular initial guess. Note that, in general, starting at different initial values yields different end points, which is not suprising given that the shown surface has 72 local optima. However, in all cases we investigated, the found optimum is

within half a per cent of the global one. In view of this, the algorithm performs very favorably and took on average only 25–30 iterations to converge.

5. Policy-driven reservoir optimization

A key objective of the research presented in this paper is to formulate policies that can be used by the autonomic self-optimizing reservoir framework to discover, select, configure, and invoke appropriate optimization services to determine optimal well locations.

The choice of optimization service depends on the size and nature of the reservoir. The SPSA algorithm studied in this paper is suited for larger reservoirs with relatively smooth characteristics. In case of reservoirs with many randomly distributed maxima and minima, the VFSA algorithm studied in our previous paper [15] can be employed during the initial optimization phase. Once convergence slows down, VFSA can be replaced by SPSA. Alternate optimization schemes (e.g., genetic algorithms, local methods such as Newton) can also be used if convergence breaks down. We plan to study and characterize the behavior and interaction of these schemes in a future work.

Similarly, policies can also be used to manage the behavior of the reservoir simulator. For example, the policy may monitor convergence of the optimizer and as it approaches the solution, it may use a finer mesh and/or smaller timesteps. The policy may even attempt to activate other numerical algorithms (e.g., time discretization schemes, solvers) or physical models (e.g., one-, two-, or three-phase flow, geomechanical). Moreover, the policy may replace IPARS by some other simulator capable of using unstructured grids or adaptive mesh refinement in order to generate more accurate simulations.

In an alternative scenario, policies may be defined to enable various optimizers to execute concurrently on dynamically acquired Grid resources, and select the best well location among these based on some metric (e.g., estimated revenue, time or cost of completion). This aspect is important for speeding up the search, or for studying the effects of parameters that were not included at the start of the optimization. For instance, some topological difficulties or unforeseen costs for drilling a well may eventually arise in some parts of the reservoir. In such a case, the expert may decide to stop the process based on a small set of nearly optimal solutions or perturb the course of the optimization (e.g. by the introduction or removal of decision variables, constraints or trial points).

The autonomic reservoir framework and the underlying Pawn peer-to-peer middleware substrate presented in this paper enable the decoupling of services and the separation of policy and mechanism. This allows external policies, such as those outlined above, to be dynamically defined and used to manage the behavior of the components/services, and to orchestrate interactions between them to achieve overall optimization goals of the reservoir.

6. Summary and conclusions

In this paper we presented the design, development, and operation of a prototype autonomic self-optimizing reservoir framework that uses peer-to-peer interactions between applications and services on the Grid to enable the autonomic optimization of well placement and operation to maximize overall revenue. The application consisted of instances of distributed multi-model, multi-block reservoir simulation components provided by IPARS, stochastic optimization services provided by SPSA, economic modeling services, real-time services providing current economic data (e.g. oil prices), archives for already computed data, and experts (scientists, engineers) connected via pervasive collaborative portals. It was built on the Pawn P2P substrate, which provided JXTA-based peer-to-peer messaging services, and the Discover computational collaboratory, which combines Grid infrastructure services provided by Globus and interaction and collaboration services. Sample outputs from the optimization process were presented that showed how the interaction of all these components can be used to solve the economically important question of where to place a new well into an existing reservoir. This problem is computationally very challenging due to the enormous complexity of optimizing a complicated mathematical model, and can benefit from the distributed and autonomous features of the approach presented here. Furthermore, the formulation of policies for the autonomic selection, configuration and invocation of optimization services are necessary ingredients of adaptively changing the components used in the optimization.

The prototype autonomic Grid application presented in this paper demonstrated the potential of the emerging Grid infrastructure and its support for secure and seamless interactions, enabling a new generation of autonomic applications. These applications will be based on peer-to-peer interactions between application components, Grid services, resources, and data, and will use separately defined policies to orchestrate these interactions and enable self-managing and self-optimizing behaviors. We believe that such autonomic behaviors will be critical for addressing the scale, complexity, heterogeneity and dynamism inherent in Grid applications and environments.

Acknowledgments

The research presented in this paper is supported in part by the National Science Foundation (NSF) via grants numbers ACI 9984357 (CAREERS), EIA 0103674 (NGS), NSF EIA-0121523/EIA-0120934 (ITR), ANI-0335244 (NRT), CNS-0305495 (NGS) and by DOE ASCI/ASAP (Caltech) via grant number 82-1052856.

References

[1] T. Arbogast, M.F. Wheeler and I. Yotov, Mixed finite elements for elliptic problems with tensor coefficients as cell-centered finite differences, SIAM J. Numer. Anal 34(2) (1997) 828–852.

[2] K. Aziz and A. Settari, *Petroleum Reservoir Simulation* (Applied Science Publishers Ltd., London, 1979).
[3] V. Bhat and M. Parashar, Discover middleware substrate for integrating services on the grid, in: *Proceedings of the 10th International Conference on High Performance Computing (HiPC 2003)*, eds. T.M. Pinkston and V.K. Prasanna, volume 2913 of Lecture Notes in Computer Science, Springer-Verlag, (Dec. 2003) pp. 373–382.
[4] A.C. Bittencourt and R.N. Horne, Reservoir development and design optimization, in: *SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, (Oct. 1997) SPE 38895.
[5] G. Chavent and J. Jaffre, *Mathematical Models and Finite Elements for Reservoir Simulation* (North-Holland, Amsterdam, 1986).
[6] I. Foster and C. Kesselman, (eds.), *Globus: A Toolkit Based Grid Architecture*, Morgan Kaufman, (1999) pp. 259–278.
[7] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure* (Morgan Kaufman, 2004).
[8] B. Guyaguler and R.N. Horne, Uncertainty assessment of well placement optimization, in: *SPE Annual Technical Conference and Exhibition* (New Orleans, Louisiana, September, 2001). SPE 71625.
[9] R. Helmig, *Multiphase Flow and Transport Processes in the Subsurface* (Springer, 1997).
[10] IPARS: Integrated Parallel Reservoir Simulator, http://www.ices.utexas.edu/CSM.
[11] S. Lacroix, Y. Vassilevski and M.F. Wheeler, Iterative solvers of the implicit parallel accurate reservoir simulator (IPARS), Numerical Linear Algebra with Applications 4 (2001) 537–549.
[12] H. Liu and M. Parashar, DIOS++: A framework for rule-based autonomic management of distributed scientific applications, in: *Proceedings of the 9th International Euro-Par Conference (Euro-Par 2003)*, eds. H. Kosch, L. Boszormenyi and H. Hellwagner, volume 2790 of Lecture Notes in Computer Science, Springer Verlag (2003), pp. 66–73.
[13] Q. Lu, M. Peszyńska and M.F. Wheeler, A parallel multi-block black-oil model in multi-model implementation, SPE Journal 7(3) (2002) 278–287, SPE 79535.
[14] V. Mann, V. Matossian, R. Muralidhar and M. Parashar, DISCOVER: An environment for Web-based interaction and steering of high-performance scientific applications, Concurrency and Computation: Practice and Experience 13(8/9) (2001) 737–754.
[15] V. Matossian, V. Bhat, M. Parashar, M. Peszynska, M. Sen, P. Stoffa and M.F. Wheeler, Concurrency and Computation: Practice and Experience, John Wiley and Sons, Vol. 17, Issue 1, pp. 1–26, 2005.
[16] R. Muralidhar and M. Parashar, A distributed object infrastructure for interaction and steering, Concurrency and Computation: Practice and Experience 15(10) (2003) 957–977.
[17] Y. Pan and R.N. Horne, Improved methods for multivariate optimization of field development scheduling and well placement design, in: *SPE Annual Technical Conference and Exhibition*, New Orleans, Louisiana, (Sept. 1998) pp. 27–30. SPE 49055.
[18] M. Parashar, G. von Laszewski, S. Verma, K. Keahey J. Gawor and N. Rehn, A CORBA commodity grid kit, Special Issue on Grid Computing Environments, Concurrency and Computation: Practice and Experience 14 (2002) 1057–1074.
[19] M. Parashar, J.A. Wheeler, G. Pope, K. Wang and P. Wang, A new generation EOS compositional reservoir simulator, Part II: Framework and multiprocessing, in: *Fourteenth SPE Symposium on Reservoir Simulation, Dallas, Texas*, Society of Petroleum Engineers (June 1997) pp. 31–38.
[20] D.W. Peaceman, *Fundamentals of Numerical Reservoir Simulation*, 1st edition (Elsevier Scientfic Publishing Company, Amsterdam-Oxford-New York, 1977).
[21] M. Peszyńska, Q. Lu and M.F. Wheeler, Multiphysics coupling of codes, in: *Computational Methods in Water Resources*, eds., L.R. Bentley, J.F. Sykes, C.A. Brebbia, W.G. Gray and G.F. Pinder, A. A. Balkema (2000) pp. 175–182.
[22] Project JXTA: `http://www.jxta.org`, 2001.
[23] T.F. Russell and M.F. Wheeler, Finite element and finite difference methods for continuous flows in porous media, in: *The Mathematics of Reservoir Simulation*, ed., R.E. Ewing, SIAM, Philadelphia (1983) pp. 35–106.
[24] M. Sen and P. Stoffa, *Global Optimization Methods in Geophysical Inversion* (Elsevier, 1995).
[25] J.C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Trans. Autom. Control 37 (1992) 332–341.
[26] J.C. Spall, Adaptive stochastic approximation by the simulateous perturbation method, IEEE Trans. Autom. Contr 45 (2000) 1839–1853.
[27] J.C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control*, Inc., Publication, John Wiley & Sons, New Jersey (2003).
[28] P. Wang, I. Yotov, M.F. Wheeler, T. Arbogast, C.N. Dawson, M. Parashar and K. Sepehrnoori, A new generation EOS compositional reservoir simulator. Part I: Formulation and discretization, in: *Fourteenth SPE Symposium on Reservoir Simulation, Dallas, Texas*, Society of Petroleum Engineers (June 1997) pp. 55–64.
[29] M.F. Wheeler and M. Peszyńska, Computational engineering and science methodologies for modeling and simulation of subsurface applications, Advances in Water Resources, in press.
[30] M.F. Wheeler, M. Peszyńska, X. Gai and O. El-Domeiri, Modeling subsurface flow on PC cluster, in: *High Performance Computing*, ed. A. Tentner, SCS (2000) pp. 318–323.
[31] M.F. Wheeler, J.A. Wheeler and M. Peszyńska, A distributed computing portal for coupling multi-physics and multiple domains in porous media, in: *Computational Methods in Water Resources*, (eds.), L.R. Bentley, J.F. Sykes, C.A. Brebbia, W.G. Gray, and G.F. Pinder, A. A. Balkema, (2000) pp. 167–174.
[32] B. Yeten, L.J. Durlofsky and K. Aziz, Optimization of nonconventional well type, location, and trajectory, SPE Journal 8(3) (2003) 200–210. SPE 86880.

Wolfgang Bangerth is a postdoctoral research fellow at both the Institute for Computational Engineering and Sciences, and the Institute for Geophyics, at the University of Texas at Austin. He obtained his Ph.D. in applied mathematics from the University of Heidelberg, Germany in 2002. He is the project leader for the deal.II finite element library (http://www.dealii.org). Wolfgang is a member of SIAM, AAAS, and ACM.
E-mail: bangerth@ices.utexas.edu

Hector Klie obtained his Ph.D. degree in Computational Science and Engineering at Rice University, 1996, he completed his Master and undergraduate degrees in Computer Science at the Simon Bolivar University, Venezuela in 1991 and 1989, respectively. Hector Klie's main research interests are in the development of efficient parallel linear and nonlinear solvers and optimization algorithms for large-scale transport and flow of porous media problems. He currently holds the position of Associate Director and Senior Research Associate in the Center for Subsurface Modeling at the Institute of Computational Science and Engineering at The University of Texas at Austin. Dr. Klie is current member of SIAM, SPE and SEG.
E-mail: klie@ices.utexas.edu

Vincent Matossian obtained a Masters in applied physics from the French Université Pierre et Marie Curie. Vincent is currently pursuing a Ph.D. degree in distributed systems at the Department of Electrical and Computer Engineering at Rutgers University under the guidance of Manish Parashar. His research interests include information discovery and ad-hoc communication paradigms in decentralized systems.
E-mail: vincentm@caip.rutgers.edu

Manish Parashar is Professor of Electrical and Computer Engineering at Rutgers University, where he also is director of the Applied Software Systems Laboratory. He received a BE degree in Electronics and Telecommunications from Bombay University, India and MS and Ph.D. degrees in Computer Engineering from Syracuse University. He has received the Rutgers Board of Trustees Award for Excellence in Research (2004–2005), NSF CAREER Award (1999) and the Enrico Fermi Scholarship from Argonne National Laboratory (1996). His research interests include autonomic computing, parallel & distributed computing (including peer-to-peer and Grid computing), scientific computing, software engineering. He is a senior member of IEEE, a member of the IEEE Computer Society Distinguished Visitor Program (2004–2007), and a member of ACM.
E-mail: parashar@caip.rutgers.edu

Mary Fanett Wheeler obtained her Ph.D. at Rice University in 1971. Her primary research interest is in the numerical solutions of partial differential systems with applications to flow in porous media, geomechanics, surface flow, and parallel computation. Her numerical work includes formulation, analysis and implementation of finite-difference/finite-element discretization schemes for nonlinear, coupled PDE's as well as domain decomposition iterative solution methods. She has directed the Center for Subsurface Modeling, The University of Texas at Austin, since its creation in 1990. Dr. Wheeler is recepient of the Ernest and Virginia Cockrell Chair in Engineering and is Professor in the Department of Aerospace Engineering & Engineering Mechanics and in the Department of Petroleum & Geosystems Engineering of The University of Texas.