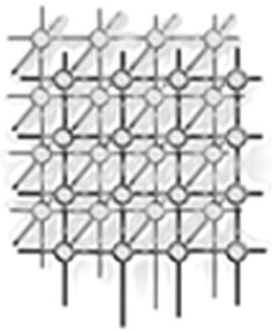
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2005; **17**:1–26
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.871

Autonomic oil reservoir optimization on the Grid



Vincent Matossian[1], Viraj Bhat[1], Manish Parashar[1,*,†], Małgorzata Peszyńska[2], Mrinal Sen[3], Paul Stoffa[3] and Mary F. Wheeler[4]

[1]*The Applied Software Systems Laboratory, Rutgers University, Piscataway, NJ 08855, U.S.A.*
[2]*Department of Mathematics, Oregon State University, Corvallis, OR 97331, U.S.A.*
[3]*Institute for Geophysics, University of Texas at Austin, Austin, TX 78759, U.S.A.*
[4]*Center for Subsurface Modeling, University of Texas at Austin, Austin, TX 78712, U.S.A.*

SUMMARY

The emerging Grid infrastructure and its support for seamless and secure interactions is enabling a new generation of autonomic applications where the application components, Grid services, resources, and data interact as peers to manage, adapt and optimize themselves and the overall application. In this paper we describe the design, development and operation of a prototype of such an application that uses peer-to-peer interactions between distributed services and data on the Grid to enable the autonomic optimization of an oil reservoir. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: autonomic computing; oil reservoir optimization; peer-to-peer; Grid services

1. INTRODUCTION

The Grid [1] is rapidly emerging as the dominant paradigm for wide area distributed computing. Its goal is to provide a service-oriented infrastructure that leverages standardized protocols and services to enable pervasive access to, and coordinated sharing of geographically distributed hardware, software, and information resources.

The Grid community and the Global Grid Forum [2] are investing considerable effort in developing and deploying standard protocols and services (e.g. the Open Grid Service Architecture [3]) that enable seamless and secure discovery, access to, and interactions among resources, services, and applications. This potential for seamless aggregation, integration, and interactions has made it

*Correspondence to: Manish Parashar, The Applied Software Systems Laboratory, Rutgers University, Piscataway, NJ, U.S.A.
†E-mail: parashar@caip.rutgers.edu

Contract/grant sponsor: NSF; contract/grant numbers: ACI 9984357 (CAREERS), EIA-0103674 (NGS) and EIA-0120934 (ITR)
Contract/grant sponsor: DOE ASCI/ASAP (Caltech); contract/grant numbers: PC295251 and 1052856

Received 14 June 2003
Revised 20 November 2003
Accepted 6 December 2003

possible for scientists and engineers to conceive a new generation of applications that enable realistic investigation of complex scientific and engineering problems. These applications will symbiotically and opportunistically combine computations, experiments, observations, and data, and will provide important insights into complex systems such as interacting black holes and neutron stars, formations of galaxies, subsurface flows in oil reservoirs and aquifers, and dynamic response of materials to detonations. However, such a global scientific investigation requires continuous, seamless and secure interactions where the application components, Grid services, resources (systems, CPUs, instruments, storage) and data (archives, sensors) can interact as peers.

In this paper we focus on one such application: the autonomic oil production management process. Such a process involves (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 that provide 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 entities 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 the current resource state, and negotiate the required quality of service. Next, we recall that the data necessary for reservoir simulation are 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 *history matching*, dynamic data injection, and data driven adaptations. Next, 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 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 sub-optimal 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 sense, the process is self-managing, self-adapting, and self-optimizing.

In this paper we describe the development and operation of a prototype application that uses peer-to-peer interactions between applications and services on the Grid which enable autonomic optimization of oil and gas recovery from a subsurface reservoir. Here we assume that the information on the properties of this reservoir is complete and adequate. The paper has three key objectives: (1) to provide a proof-of-concept implementation to evaluate the correctness and feasibility of the formulation and methodology, (2) to use this sample application to demonstrate the feasibility and benefits of such self-optimizing applications, and (3) to develop a prototype peer-to-peer (P2P) middleware substrate that provides the functionalities required to enable the identified application interactions. The prototype

application optimizes the placement and operating conditions of oil wells to maximize the overall revenue from production. The application consists of (i) instances of distributed multi-model, multi-block reservoir simulation components provided by the Integrated Parallel Accurate Reservoir Simulator (IPARS) reservoir simulator framework, (ii) optimization services provided by an algorithm called Very Fast Simulated Annealing (VFSA) based on *simulated annealing*, (iii) economic modeling services, (iv) real-time services providing current economic data (e.g. oil prices), (v) historical data archives, and (vi) experts (scientists, engineers) connected via pervasive collaborative portals. It is built on the Pawn P2P substrate, which provides JXTA-based [4] P2P messaging services, and the Discover computational collaboratory, which combines Grid infrastructure services provided by Globus [5] and interaction and collaboration services.

The rest of this paper is organized as follows. Section 2 describes the application and introduces the underlying models and components. Section 3 describes the design and implementation of the peer services that are part of the application. Section 4 presents an overview of the Pawn interaction middleware. Sections 5 describe the overall operation of the application. Section 6 presents a summary and conclusions.

2. AUTONOMIC OIL RESERVOIR OPTIMIZATION: PROBLEM DESCRIPTION

In this section we present an overview of the prototype application which is the management and optimization of the oil production process. We specify the mathematical models underlying the reservoir simulation and the economic model, introduce the case study, and describe the optimization algorithm.

Assume that there exists an oil reservoir whose properties are known, at least, at a given scale, and in which a few wells are operating but it is known that more wells need to be drilled. The objective of the management process is to find optimal placement and operating conditions of new (and existing) wells which are elements of the parameter space P. The notion of 'optimality', of course, is tied to the definition of an objective function. Once the optimality criteria are fixed and the parameter space P is defined, an optimization algorithm is applied to find p_{opt} such that $f(p_{\text{opt}})$ satisfies given criteria, for example, is minimal. Here $f(\cdot)$ can be any continuous but not necessarily differentiable function. Therefore, we focus on a class of non-gradient based optimization methods which require only the evaluation of the objective function $f(p)$ and not of its gradient.

Below we give a brief overview of the mathematical models defining P and $f(p)$, $p \in P$, as well as the optimization method which determines p_{opt}. These are essentially independent of one another, that is, the definition of P or $f(P)$ can be changed independently of the optimization algorithm. This independence is reflected in the implementation by the fact that the reservoir simulation model which delivers $f(p)$ for a given p is a separate application (here, an IPARS instance initiated by the IPARS factory), which interacts with the optimization service (here, implementing the VFSA algorithm) using the Pawn interaction middleware.

2.1. Mathematical model for the flow in an oil reservoir

We consider a heterogeneous 3D oil reservoir, denoted by Ω, surrounded by impermeable rocks. The flow in the reservoir is assumed to be of two-phase immiscible character [6]. Considered are two

components (hydrocarbon and aqueous) which do not mix and which are identified with their phases. Partial differential equations which describe the conservation of mass of each component $m = \mathrm{o}, \mathrm{w}$, respectively, oil and water, are

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

where the fluxes U_m are defined using the multiphase form of Darcy's law [7] which, if gravity is ignored, reads $U_m = -\rho_m K \lambda_m \nabla P_m$. In these equations, ϕ is the porosity of the porous medium, K denotes permeability tensor, ρ_m denotes density of a component, λ_m denotes mobility of a component, N_m denotes concentration of a component m, and q_m denotes sources (production and injection rates). Additional equations specifying volume, capillary, and state constraints are added, and boundary and initial conditions complement the system—see [6,7]. Finally, $N_m = S_m \rho_m$ with S_m denoting saturation of a phase. The resulting system, shown here with gravity ignored, reads

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

In this paper we consider oil and water production wells and water injection wells. At an injection well, q_w is non-negative (we will use the notation $q_\mathrm{w}^+ := q_\mathrm{w}$ to make it explicit). At a production well, both q_o and q_w may be non-zero and if so, they are non-positive and we will denote this by $q_m^- := -q_m$ at a production well. The injection rates are determined from the input, either explicitly by specifying the volume or mass rate of a component or implicitly using the numerical model by specifying the pressure in the well using the so-called *Bottom Hole Pressure* (BHP) parameter. In particular, for a production well to be active, BHP at that well must be lower than the surrounding reservoir pressure. Similarly, BHP at an injection well must exceed the reservoir pressure. Subtle details regarding multiphase properties of fluids injected and produced apply but will not be discussed here.

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 [8,9]. Time discretization can be either fully implicit, semi-implicit or sequential, but here we only consider the sequential method in which there are two linear systems of equations: the pressure equation and the saturation equation, which are solved at each time step. The latter equation, formulated using the *total velocity concept*, may have its own time-stepping which is adjusted according to the stability requirements. Details of this model and its formulation are available in [10]. Stability of the sequential model as opposed to explicit models is discussed in [6].

We note that in the examples discussed in this paper the (saturation) time step should be, in the purely hyperbolic case (no capillary pressure), not much larger than around $\frac{1}{3}$ day, which follows from calculations of stability criteria applied to the data of the problem. Practically, however, due to non-trivial diffusive effects arising from the capillary and numerical diffusion, a time step as large as 1 day leads to stable well rates. This time step is used uniformly in all evaluations of $f(p)$ for different p.

2.2. 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

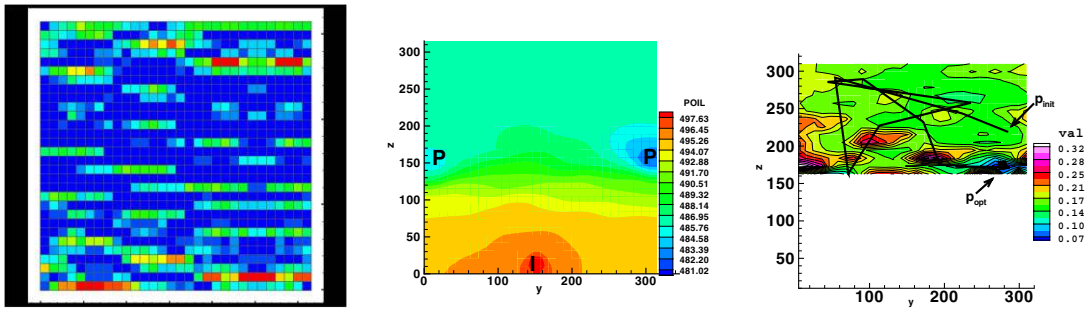


Figure 1. Case study. 2D view of the reservoir. Left: permeability field. Center: the pressure solution which is computed with two production wells (indicated by letters 'P' and associated with low pressure) and one injection well (indicated by the letter 'I' and associated with high-pressure values). Right: contours of the objective function $f(p)$ over the parameter space $p \in P$. P is the upper half of the reservoir where a new well is drilled. The minimum of $f(p)$ corresponds to the optimal position of the new well p_{opt}. It is found by the VFSA algorithm; see the random walk over P starting at p_{init} which converges to p_{opt}.

oil, costs of injection, extraction, and disposal of water and of the hydrocarbons, as well as associated operating costs. Appropriate constants are denoted, respectively, by P_{drill}, p_{o}, $p_{\text{w,disp}}$, $p_{\text{w,inj}}$, p_{op}.

For the model considered in this paper, the economic value of production or total revenue from a field can be computed as

$$E_{\text{val}}(t) = \sum_{\text{wells}^-} \int_0^t (p_{\text{o}} q_{\text{o}}^-(s) - p_{\text{w,disp}} q_{\text{w}}^-(s))\,\mathrm{d}s - \sum_{\text{wells}^-} \int_0^t p_{\text{op}}(q_{\text{w}}^-(s) + q_{\text{o}}^-(s))\,\mathrm{d}s - \sum_{\text{wells}^+} \int_0^t p_{\text{w,inj}} q_{\text{w}}^+(s)\,\mathrm{d}s - P_{\text{drill}} \tag{3}$$

In this numerical model, all integrals are computed using a left-rectangular quadrature rule. Additionally, a discount (percentage rate) r is applied. The implementation of the model is presented in Section 3.4.

2.3. Case study

In our case study we consider a 3D reservoir Ω with a spatial grid chosen so that

$$\Omega = 3x3' \times 30x10.5' \times 30x10.5'.$$

We note that the gravity direction is along the $x(i)$ direction. The porosity is fixed and $\phi \equiv 0.2$ but, the reservoir has a heterogeneous permeability field as shown in Figure 1, left. The fluids are initially in equilibrium.

In this reservoir, three wells—one injection well and two production wells—have been placed. See the resulting oil pressure contours in Figure 1, right, where high-pressure values indicate an injection well and low-pressure values a production well. In order to displace the oil residing in the

upper half of the field, it is necessary to place an additional injection well, which naturally should be located in the upper half of the reservoir.

We define the parameter space $P = \{(y, z)\}$, where a pair $p = (y, z) \in (0, 315) \times (157.5, 315)$ defines the aerial position of a well in the upper half of the reservoir. We assume that the well penetrates through the entire depth of the reservoir, that is, the x coordinates of its bottom and top are fixed. We also fix the BHP operating conditions at the new injection well to be the same as the BHP at the other injection well located in the center of the bottom half of Ω. 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.

Assume that we want to find the position of a new well which is optimal in some sense. For example, an optimal position of a well would be such that it maximizes the overall economic value of production $E_{\text{val}}(p)$ defined by Equation (3). Another possible objective function could be the amount of bypassed oil (i.e. oil left in the field after production) which would be minimized. In addition, one could place a well in such a location that the amount of water being produced from production wells would be minimal. This last case is somewhat akin to preventing the *water coning* and the *water fingering* phenomena [6,11]. Note that the (negative) cost of water produced from the production wells appears in the objective function defined by Equation (3). However, it is only one of its components. In summary, in this paper we do not consider objective functions other than the one following from Equation (3).

Due to the numerical well model that we used, the function $E_{\text{val}}(p)$ is piecewise constant over $p = (y, z)$ in each discretization cell in Ω. Therefore, for our purposes,

$$\bar{P} = \{(y_j, z_k),\ j = 1 \ldots 30, k = 16 \ldots 30, \quad y_j = 10.5 * (j - \tfrac{1}{2}),\ z_k = 10.5 * (k - \tfrac{1}{2})\}$$

represents P well.

Theoretically, one can find the values of $E_{\text{val}}(p)$, $p \in \bar{P}$ and select p_{opt} out of 450 members of the set $\bar{P}$ in a straightforward manner. The contour plot of $f(p)$ shown in Figure 1 is a result of such a procedure. Such a *direct* method, however, is computationally very expensive since for each p, evaluating $E_{\text{val}}(p)$ requires solving a full 3D reservoir simulation problem for each of the 450 cases.

As a result, it is mandatory to find a fast and reliable optimization algorithm which does not require evaluating $E_{\text{val}}(p)$ over the entire set $p \in \bar{P}$. In this paper we achieve this using the VFSA algorithm discussed below.

2.4. Optimization algorithm: VFSA

Simulated annealing (SA) is analogous to the natural process of crystal annealing when a liquid gradually cools to a solid state. The SA technique starts with an initial model $p_0 \in P$, with associated cost or energy $f(p_0)$. It then draws a new model $p_1 \in P$ from a flat distribution of models within P. The associated energy $f(p_1)$ for the new model is then computed, and compared against the initial model. If the energy of the new state is less than the initial state, the new state is considered to be good and is accepted to replace the initial model unconditionally. However, if the energy of the new state is larger than the initial state, the new model is accepted with a defined probability. This process is repeated a large number of times, with the annealing temperature gradually decreasing according to the predefined scheme. With a carefully defined cooling schedule, a global minimum can be found. One may choose a linear or logarithmic decreasing cooling scheme. The tradeoff here is between the computation cost and the accuracy of the result. Fast cooling will fail to produce a crystal while slow cooling takes a long time.

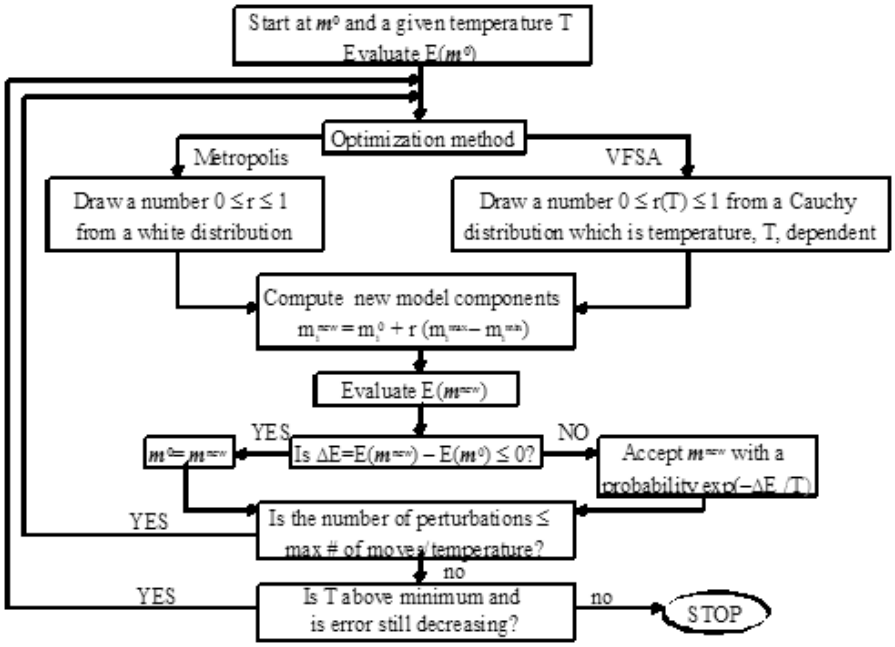


Figure 2. An overview of VFSA.

VFSA is a variant of SA that speeds up the annealing process. An overview of the VFSA algorithm is given in Figure 2. VFSA differs from SA in the following ways. The new model is drawn from a temperature-dependent Cauchy-like distribution centered around the current model. This change has two fundamental effects. First, it allows for larger sampling of the model space at the early stages of the inversion when the temperature is high, and much narrower sampling in the model space as the inversion converges when the temperature decreases. Second, each model parameter, that is, each element of the parameter space P, can have its own cooling schedule and model space sampling scheme. For example, in our case this concerns y and z, which can have their own cooling schedules and sampling schemes. Therefore, it allows for the individual control for each parameter, and the incorporation of *a priori* information. Applications of VFSA to several geophysical inverse problems can be found in [12].

The application of VFSA to finding $p_{\text{opt}} \in P$ which maximizes $E_{\text{val}}(p)$ requires that we define the 'energy of the system' by normalizing $E_{\text{val}}(p)$ so that the maximum of $E_{\text{val}}(p)$ will correspond to the minimum of the 'energy'. This can be done by a simple scaling map $E_{\text{val}}(p) \mapsto f(p)$ termed as the 'normalizing procedure' below.

In our case study, as shown in Figure 1, the VFSA algorithm starts from a random initial position p_{init} and proceeds through a sequence of 'random walks' over P to finally settle on the solution p_{opt}

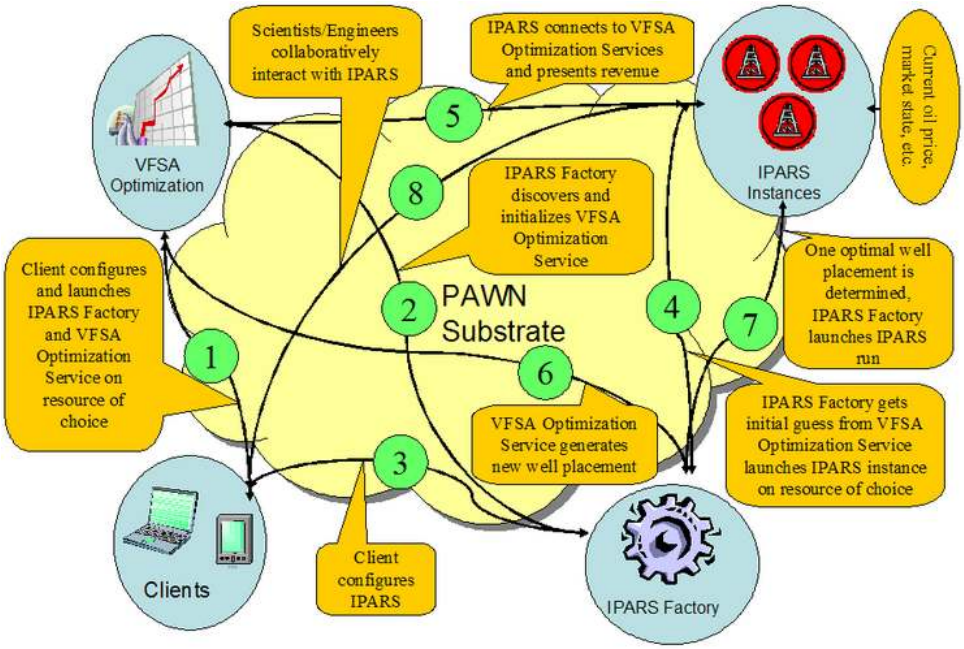


Figure 3. Autonomous oil reservoir optimization using decentralized services.

which falls at the minimum of $f(p)$, $p \in P$ equivalent to the one determined by the direct method. In the algorithm, the walk proceeds over P but the evaluations are only made for members of its finite dimensional projection $\bar{P}$, if necessary. In other words, consider two different elements of the parameter space $p_1 \in P$, $p_2 \in P$, $p_1 \neq p_2$ which have the same projections $\bar{p}_1 = \bar{p}_2 \in \bar{P}$. In the algorithm, every value $E_{\text{val}}(p)$ that has been evaluated is archived in a database. Therefore, if $f(p_1)$ exists in the database, then there is no need to evaluate $f(p_2)$ which, by definition, can be read from the database. In the example shown in Figure 1, the random walk over the parameter space P is executed. The walk proceeds over 81 elements of P but only 20 evaluations, that is, of $p \in \bar{P}$, are necessary.

3. ENABLING AUTONOMIC OIL RESERVOIR OPTIMIZATION USING DECENTRALIZED SERVICES

The overall application scenario is illustrated in Figure 3. The primary peers and services participating in the application are described below.

3.1. Integrated Parallel Accurate Reservoir Simulator (IPARS)

IPARS [13–20] is a parallel reservoir simulation framework for modeling multiphase, multiphysics flow in porous media. IPARS offers sophisticated simulation components that encapsulate complex mathematical models of the physical interaction in the subsurface, such as the one described in Section 2, and which execute on parallel and distributed systems.

IPARS provides key infrastructure services including memory management, well management, message passing, and input/output management. Solvers used by IPARS employ state-of-the-art techniques for nonlinear and linear problems, including multigrid and other preconditioners [21]. An arbitrary number of wells, each with one or more completion intervals, is supported. Furthermore, the framework supports multiple physical models and their multiphysics couplings [19,20,22]. Models can be coupled with one another and with external applications [23]. IPARS is primarily implemented in Fortran and C and is integrated with the application framework 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 selecting appropriate models from those provided by IPARS, defining the structure and properties of the reservoir to be simulated, specifying required parameters and producing relevant input files. Deployment and management of IPARS instances uses services provided by Discover [24] and Globus [5], and build on the CORBACoG Kit [25].

3.3. VFSA Optimization service

The VFSA Optimization service runs on the Optimization peer and implements the VFSA 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 VFSA algorithm to generate guesses of new well positions based on the normalized economic value returned by an IPARS run. The well positions are normalized to match the discretization of the IPARS reservoir simulation model. The guess is then compared with an archive of guesses, therefore preventing a guess from being repeated as described in Section 2.4. If no match is found, 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 2.2 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. In our case study, we consider a fixed reservoir with a fixed number of wells and compare the economic value obtained for the same number of wells but with different locations. Also, only two components are present (oil and water) and only water can be injected but both water and oil can be produced.

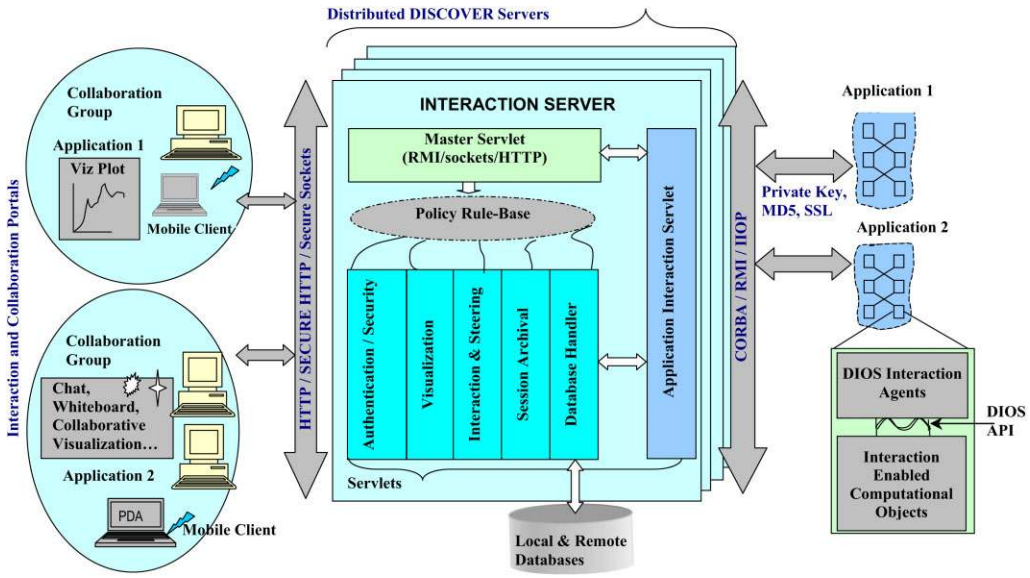


Figure 4. Architectural schematic of the Discover computational collaboratory.

A more general model incorporating gas production and injection can be formulated but will not be discussed here.

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 per volume injected 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, one may be able to use a set of 'forecasts' of prices, delivered by stochastic or other mathematical models, instead of fixed prices obtained by the network information services. This capability is not currently implemented, however, and is not a part of the prototype application.

3.5. Discover computational collaboratory

Discover [24] 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' desktop, enabling them to collaboratively access, interrogate, interact with,

and steer these applications using pervasive portals. The architecture of the Discover collaboratory is shown in Figure 4. Key components of the Discover collaboratory include the following.

- *Discover interaction and collaboration middleware substrate* [26,27] 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.
  The Discover middleware also integrates Discover collaboratory services with the Grid services provided by the Globus Toolkit [5] using the CORBA Commodity Grid (CORBA CoG) Kit [25]. 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 to use Discover to connect to and collaboratively monitor, interact with, and steer the applications.
- *DIOS Interactive Object Framework (DIOS)* [28,29] 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* [24] 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.

Using the Discover computational collaboratory clients can connect to a local server using the portal. They can also 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 3, the experts use the portals to interact with the Discover middleware and the Globus Grid services to discover and allocate appropriate resources, and to deploy the IPARS Factory, VFSA and Economic Model peers (step 1). The IPARS Factory discovers and interacts with the VFSA service peer to configure and initialize it (step 2). The expert interacts with the IPARS Factory and VFSA 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

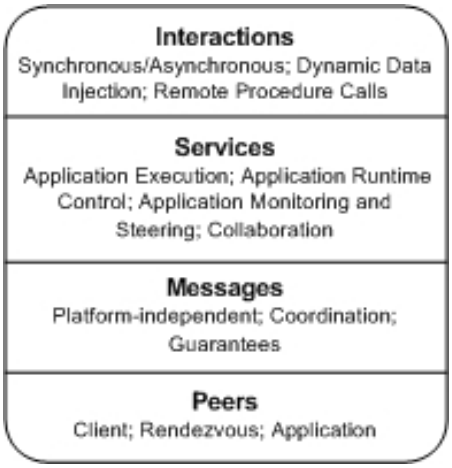


Figure 5. Conceptual design overview of Pawn.

now interacts with the Economic Model to determine current revenues, and discovers and interacts with the VFSA service when it needs optimization (step 5). VFSA provides the IPARS Factory with optimized well information (step 6), which is then used to configure and launch new IPARS simulations (step 7). Experts can, at any time, 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.

4. DESIGN AND IMPLEMENTATION OF PAWN

A conceptual overview of the Pawn P2P substrate is presented in Figure 5 and is composed of peers (computing, storage, or user peers), network and interaction services, and mechanisms. These components are layered to represent the requirements stack enabling interactions in a Grid environment. The figure can be read from bottom to top as 'Peers compose messages handled by services through specific interaction modalities'.

The overall architecture of Pawn is presented in Figure 6. It presents peergroups, composed of peers, that build on the Grid fabric and implement Pawn services. The Pawn framework is also presented; it is composed of interaction types and services, which are described in this section.

4.1. Pawn services

A network service is a functionality that can be implemented by a peer and made available to a peergroup. File-sharing or printing are typical examples of network services. In Pawn, network services are application-centric and provide the mechanisms to query, respond, subscribe, or publish

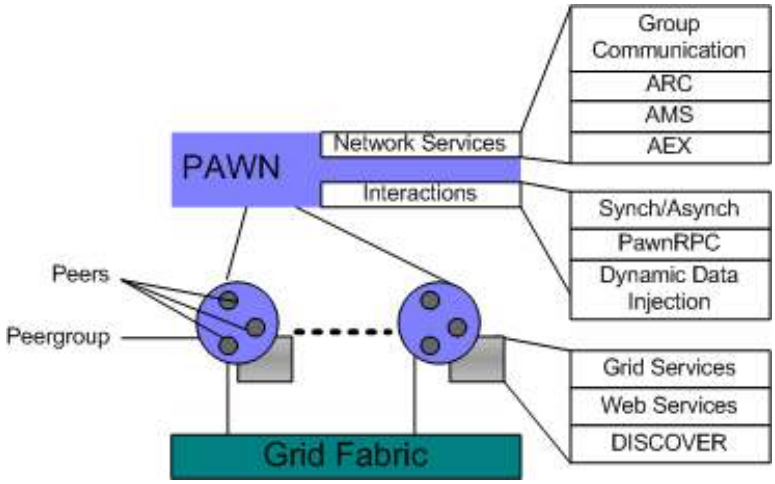


Figure 6. Pawn architecture: Pawn builds on network and interaction services to enable P2P interactions for applications on the Grid.

information to a peergroup. Pawn offers four key services to enable dynamic collaborations and autonomic interactions in scientific computing environments. These services are *Application Runtime Control*, *Application Monitoring and Steering*, *Application Execution*, and *Group Communication*. These services are briefly described below along with the support they require from the Pawn P2P messaging substrate.

4.1.1. Application Execution service

The Application Execution (AEX) service enables a peer to remotely start, stop, get the status of, or restart an application. In current systems, users require an account on the domain where an application is hosted. A new user wishing to access the application will therefore need a local account to access the compute resources on a certain domain. This task is still performed manually by the system administrator and is not suitable for a dynamic environment in which users may require short time access to resources. The AEX service offers control of an application execution dynamically without the need for a user to obtain an account on the domain where the application resides. This is achieved by providing access to users through the Pawn framework that is running on every domain as an authenticated and valid user. 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.

4.1.2. Application Monitoring and Steering service

When interacting with an application, users generally monitor the evolution of the application, they may query the application, or inject new data into the application to collect results. The Application

Table I. Peer types and associated services.

Services	AEX	ARC	AMS	Group communication	Routing
Client	No	No	No	Yes	Depends
Rendezvous	No	No	No	No	Yes
Application	Yes	Yes	Depends	No	Depends

Monitoring and Steering (AMS) service handles interactive (Request/Response) application querying (i.e. PULL), and dynamic steering (i.e. PUSH) of application parameters. It requires support for synchronous and asynchronous communications with message delivery guarantees, and for dynamic data injection (e.g. to push information to an application at runtime).

4.1.3. Application Runtime Control service

Upon starting an application, users should be notified of the existence, location, and communication modalities of the application in order to start interacting with it. The Application Runtime Control (ARC) service announces the existence of an application to the peergroup, sends application responses, publishes application update messages, and notifies the peergroup of an application termination. This service requires mechanisms for pushing information to the peergroup and for responding to queries.

4.1.4. Collaboration service (Group Communication, Presence)

The Collaboration service provides collaborative tools and support for group communication and detection of presence. Users interested in an application can monitor and steer applications, and require mechanisms to control the collaborative interaction process. In addition to interacting with applications, users can interact with each other to exchange ideas and share points of view based on display of results obtained from the applications runs. Collaborating peers need to establish direct end-to-end communications through synchronous/asynchronous channels (e.g. for file transfer or text communication), and be able to publish information to the peergroup (i.e. Transaction and PULL interactions).

4.2. Types of peers in Pawn

In Pawn, peers can implement one or more services (i.e. behaviors). The combination of services implemented by a peer defines its role. Typical roles for a peer are client, application or rendezvous and are presented in Table I; the services are shown horizontally and the types of peers vertically. 'No' signifies that the peer does not implement the given service; 'Yes' indicates that the peer does implement the given service; finally 'depends' indicates that the peer may or may not implement this service.

Every peer maintains a cache where advertisements are stored; the cache is updated upon receiving new advertisements. If the discovered advertisement was not already present in the cache, it is then added; if it was present in the cache, it is discarded.

4.2.1. Client peer

The client peer 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. Upon joining the group, a client peer advertises its presence through a presence advertisement (see Figure 7(a)), other peers can then discover the newly joined peer and start communicating with it. This peer advertisement contains the peer identifier that uniquely identifies it across all groups, the peergroup identifiers of which the peer is a member, a list of service parameters that are peer services implemented by this peer, an email address used to validate the authenticity of the peer for Chat communications, and a presence status stating the network status of this peer to the group (i.e. online, offline, away). When leaving the group, a peer publishes a presence advertisement informing other peers that its status has changed to offline. The joining procedure is typically a connection to a rendezvous peer that is part of the group. JXTA provides a mechanism to configure a peer with a list of predefined rendezvous endpoint addresses. We used this feature to direct every peer to a Web link listing all rendezvous peers which are part of the Pawn framework. Upon startup, peers download the list of all rendezvous peers and can proceed to connect to the network through a rendezvous peer. A client peer may act as a rendezvous peer either from an initial configuration or switch to this behavior at runtime.

4.2.2. Application peer

An application peer exports its 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 [24]; in such a case, the application peer acts as a proxy peer, relaying queries and responses to and from clients to applications. Application peers publish an application advertisement (see Figure 7(b)) that informs the group about the application name, the peer identifier on which it is currently running and gives a brief description of the application, specifies an application type (e.g. simulation, remote startup utility, Web service, open Grid service, etc.), and contains a handler name that every subsequent communication with this specific application will need to use as a handler tag. When newly joined peers are discovered, an application advertisement is sent to the group to inform all peers about the existence of the application.

4.2.3. Rendezvous peer

The role of the rendezvous peer is to distribute or relay messages. Rendezvous peers filter messages as defined by filtering rules input from the connected clients. Rendezvous peers route messages from a source to a destination—reliable TCP unicast messages traveling between specific endpoint addresses to an unreliable group multicast. The rendezvous peer is a lightweight component that can fail without incurring global system failure. This is possible as long as there are enough rendezvous peers deployed in a system, allowing peers to reconnect to another rendezvous peer whenever a rendezvous

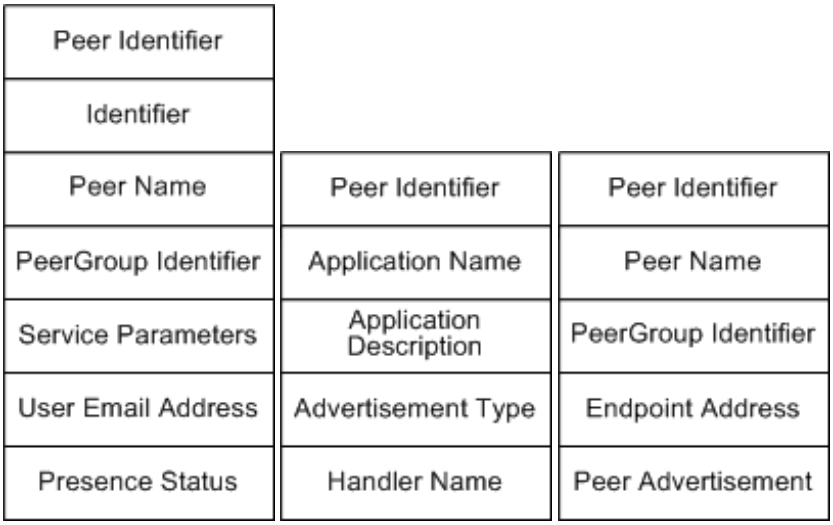


Figure 7. (a) Metadata of a peer advertisement; (b) metadata of an application advertisement; (c) metadata of a rendezvous advertisement.

disconnects or fails. Upon joining the network, rendezvous peers advertise their existence by publishing a rendezvous advertisement (see Figure 7(c)) that other peers can use to initiate a connection to the rendezvous. A rendezvous advertisement contains the unique peer identifier, the name of the peer and the identifier of the group. We modified the original JXTA implementation of the rendezvous advertisement to include the endpoint address as well as the peer advertisement of the physical node on which the rendezvous is running in order for the peers discovering the advertisement to initiate a rendezvous connection at runtime.

4.3. Implementation of Pawn services

JXTA defines unicast pipes that provide a communication channel between two endpoints, and propagate pipes that can multicast a message to a peergroup. It also defines the resolver service that sends and receives messages in an asynchronous manner. The recipient of the message can be a specific peer or a peergroup. The pipe and resolver services transfer messages using available underlying transport protocols (TCP, HTTP, TLS). To implement the four services identified above, Pawn extends the pipe and resolver services to provide stateful and guaranteed messaging. This messaging is then used to enable the key application-level interactions such as synchronous/asynchronous communication, dynamic data injection, and remote procedure calls.

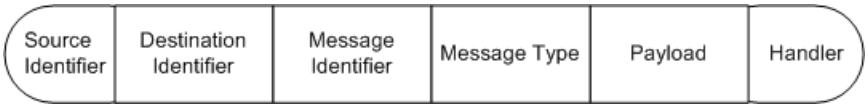


Figure 8. Format of a Pawn message.

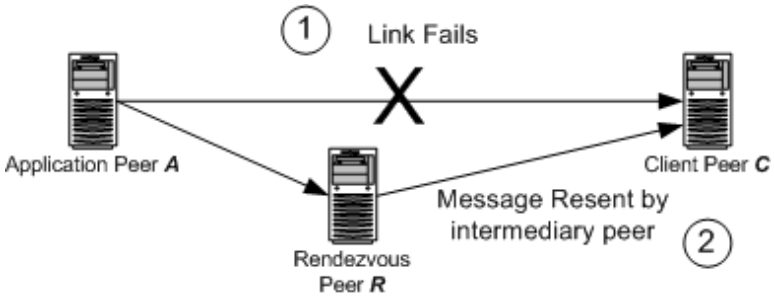


Figure 9. Using stateful messages, after detection of a transmission failure, a message can be resent from an intermediary peer without the need to be recomposed by its original sender.

4.3.1. Stateful messages

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; this message format is shown on Figure 8. The handler tag uniquely identifies the service that will process the message. 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. Intermediary peers can act as proxies, and handle storing and forwarding of messages to and from a peer. Figure 9 shows the sequence of events following a link failure, which is detected if an acknowledgement to a message is not received from the destination peer. Acknowledgment messages are received from the intermediary proxy peer as well as the sender peer; if an acknowledgement is not received, the proxy peer will forward messages on behalf of the sender peer. This capability is combined with the monitoring of the status of the peer connected to the proxy peer. If the peer is down, the proxy will act on behalf of the peer for a determined time set by a timeout value determined during the connection phase. Figure 9 shows such an event in which, in response to a peer failure, an intermediary rendezvous node takes care of re-sending a message that was meant to be sent before the sender peer failed. The failure detection is handled by the rendezvous peer monitoring the status of the peers connected to it. Once a failure is detected, the rendezvous will send every message from its queue that is labeled with the failed peer identifier. It will store all the responses for the failing peer for the determined timeout value and forward these responses to the peer if it comes back online (see Figure 10).

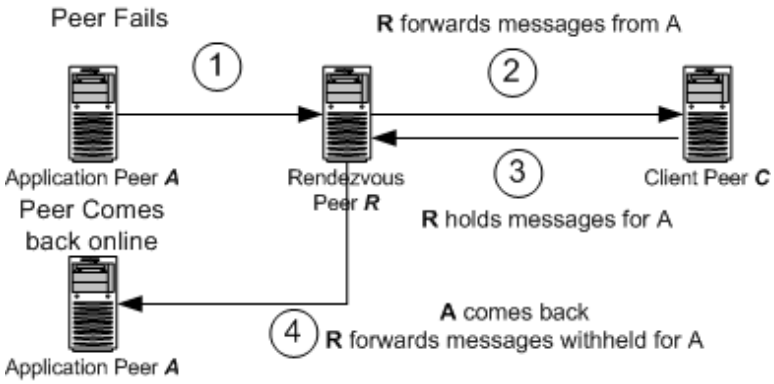


Figure 10. After detection of a peer failure, messages can be retrieved from a proxy peer once the peer comes back up. Message ordering and payload information are contained within the stateful message and do not require message recomposition from the sender.

In addition, messages can include system and application parameters in the payload to maintain application state.

4.3.2. Message guarantees

Pawn implements application-level communication guarantees by combining stateful messages and a per-message acknowledgment table maintained at every peer. FIFO message queues are used to handle all incoming and outgoing messages. Every outgoing message that expects a response is flagged in the table as awaiting acknowledgment. This flag is removed once the message is acknowledged. Messages contain a default timeout value representing an upper limit on the estimated response time. If an acknowledgment is not received and the timeout value expires, the message is resent by an intermediary node. The message identifier is a composition of the destination and sender's unique peer identifiers. It is incremented for every transaction during a session (interval between a peer joining and leaving a peergroup) to provide application-level message ordering guarantees.

4.3.3. 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 a guarantee mechanism. Figure 11 presents a request/response interaction between the AMS and ARC services, and the message queuing for outgoing and incoming messages during an end-to-end communication. The figure shows that a message (query) is formed and added to the outgoing queue by the peer implementing AMS. This message is then sent out to the peer implementing ARC. The receiving peer places the message in a queue before processing it to maintain ordering of application-level messages. Once the message is processed, a similar mechanism in the reverse direction is used to send back a response to

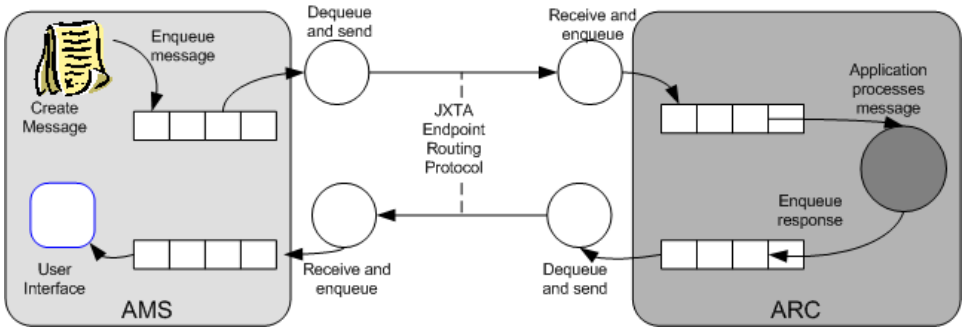


Figure 11. An interaction between a peer implementing the AMS service and a peer implementing the ARC service. Messages are added to the queue to be sent in order, and are received and enqueued to be processed in order.

the requesting peer. When using synchronous communication, the sender blocks its processing after adding the message to the queue, until reception of the corresponding response.

4.3.4. Dynamic data injection

Pawn leverages the JXTA pipe mechanism and combines it with its guaranteed message delivery mechanism to provide dynamic data injection. The pipe advertisement is obtained from the peer advertisement published by every peer in Pawn; the pipe allows unique identification of an input and output communication channel to the peer. This pipe is used by other peers to create an end-to-end channel to dynamically send and receive messages.

Every interacting peer implements a message handler that listens for incoming messages on the peer's input pipe channel. The message payload is passed to the application/service identified by the handler tag field, allowing for data to be dynamically injected into the application at runtime. This process enables applications to build autonomic interactions that require very limited human intervention; for example, services can dynamically inject data into a running application process until a certain threshold value is reached.

4.3.5. Remote procedure calls (PawnRPC)

The PawnRPC mechanism provides the low-level constructs for building application 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. The interfaces for the methods are exported by a peer and are published as part of the peer advertisement during peer discovery.

Figure 12 presents the sequence of operations for a PawnRPC call. The figure shows two distinct sequences, one labeled by the letters A–B–C and the other labeled by the numbers 1–6. The A–B–C sequence represents the operations prior to the remote invocation call. These involve registering the object that will receive method calls (A), advertising the interfaces of this object to the group (B), and

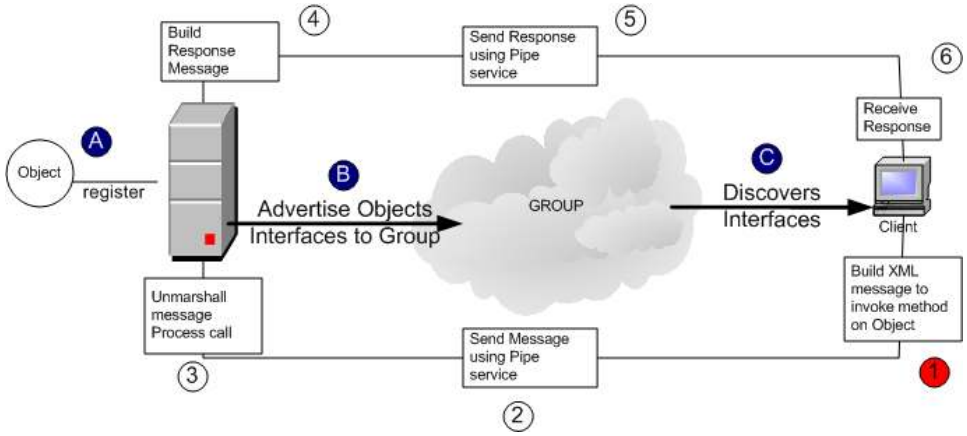


Figure 12. Sequence of operations for a PawnRPC call on a dynamically registered object.

a remote peer discovering this advertisement (C). The 1–6 sequence presents the successive operations required to invoke a remote procedure call and obtain a response. (1) The PawnRPC XML message is a composition of the destination address, the remote method name, the arguments of the method, and the argument-associated types. (2)(3) Upon receiving a PawnRPC message, a peer locally checks the credentials of the sender, and if the sender is authorized, (4) the peer invokes the appropriate method and (5)(6) returns a response to the requesting peer. The process may be done in a synchronous or asynchronous manner. PawnRPC uses the messaging guarantees to assure delivery ordering, and stateful messages tolerate failure.

5. 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, VFSA, 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 VFSA), peer initialization and configuration (e.g. Expert configures VFSA), autonomic optimization (e.g IPARS and VFSA 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.

5.1. IPARS Factory and VFSA Optimization service deployment

The IPARS Factory and VFSA Optimization peers are deployed using Globus services accessed through Discover/CORBACoG. The VFSA peer is a Fortran program with C wrappers and is integrated

Figure 13. Peer deployment.

with Pawn using the Java Native Interface. Figure 13 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 MDS to which the Expert has access privileges. Authentication and authorization is based on the Globus GSI service. Once authenticated, the Expert can use the portal to deploy the IPARS Factory and VFSA 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.

5.2. Peer initialization and discovery

The discovery process between the IPARS Factory and the VFSA Optimization peers is illustrated in Figure 14. At startup, peers use the underlying JXTA discovery service to publish an advertisement to the peergroup. This advertisement 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 VFSA service.

5.3. IPARS and VFSA configuration

The Expert uses the portal and the control interfaces exported to configure the VFSA 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 setup IPARS instances during the optimization process, and initialize the VFSA 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 VFSA peer. A response is generated and sent back (using AEX) to the client to confirm the configuration change.

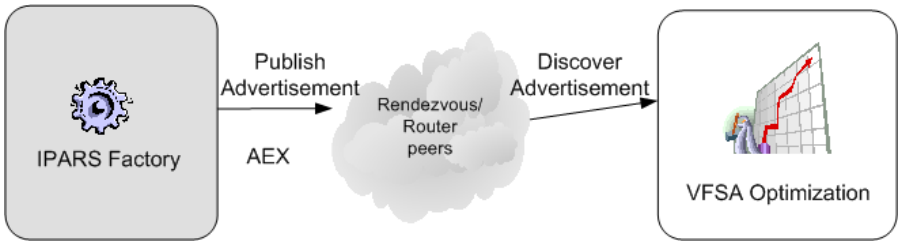


Figure 14. Peer discovery.

5.4. 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, VFSA 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 $E_{\text{val}}(p)$ for the current guess p_{current}. This revenue is normalized and then communicated as $f(p_{\text{current}})$ to the VFSA. The VFSA uses this value to generate an updated guess of the well parameters p_{new} and sends this 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 (e.g. revenue stabilizes). Figure 15 shows the overall optimization process between IPARS Factory, IPARS, and VFSA. Note that Experts can connect to any of these peers at any time and steer the optimization process.
- *Well parameter and normalized revenue archive*. After each iteration of the optimization process, normalized well parameters produced by VFSA, and the revenue and normalized revenue produced by the Economic Model are stored in an archive (a MySQL database) maintained by an archival peer. During the optimization process, when a new set of well parameters is received from VFSA, the IPARS Factory checks the archive before launching an IPARS instance. If the current guess is already present in the archive, the corresponding normalized revenue value is sent to VFSA 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 client–server system, but is implemented in a purely P2P manner.

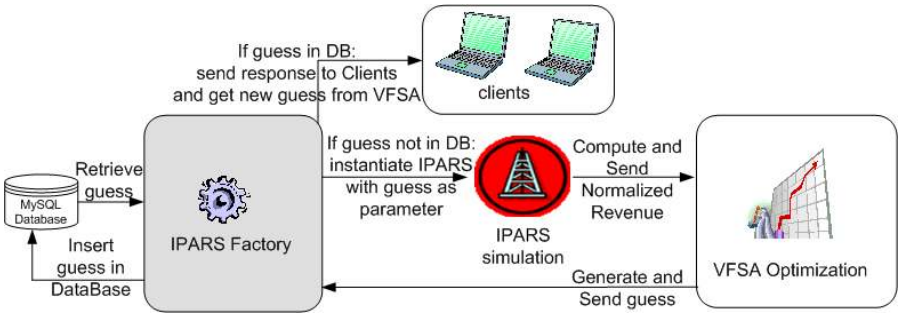


Figure 15. Optimization process.

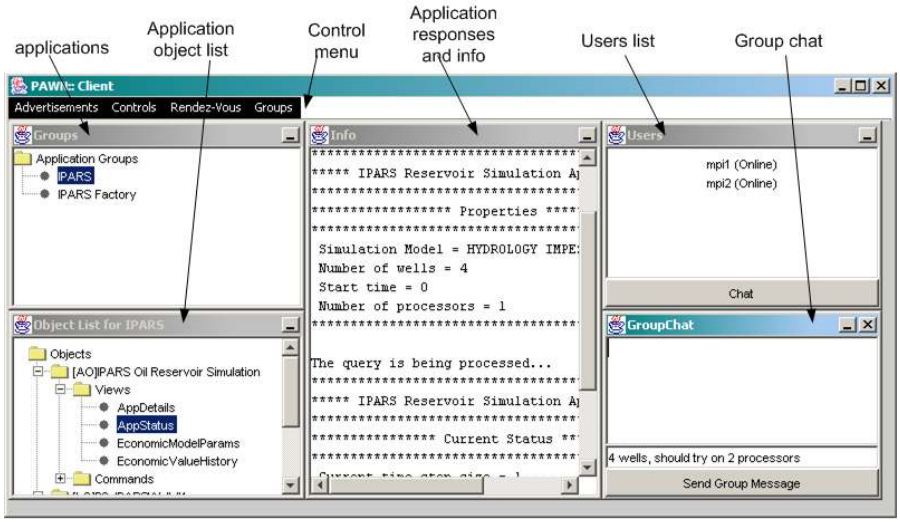


Figure 16. Graphical user interface of the expert's portal.

5.5. 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 parameters, 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 16 presents the client peer's portal interface used

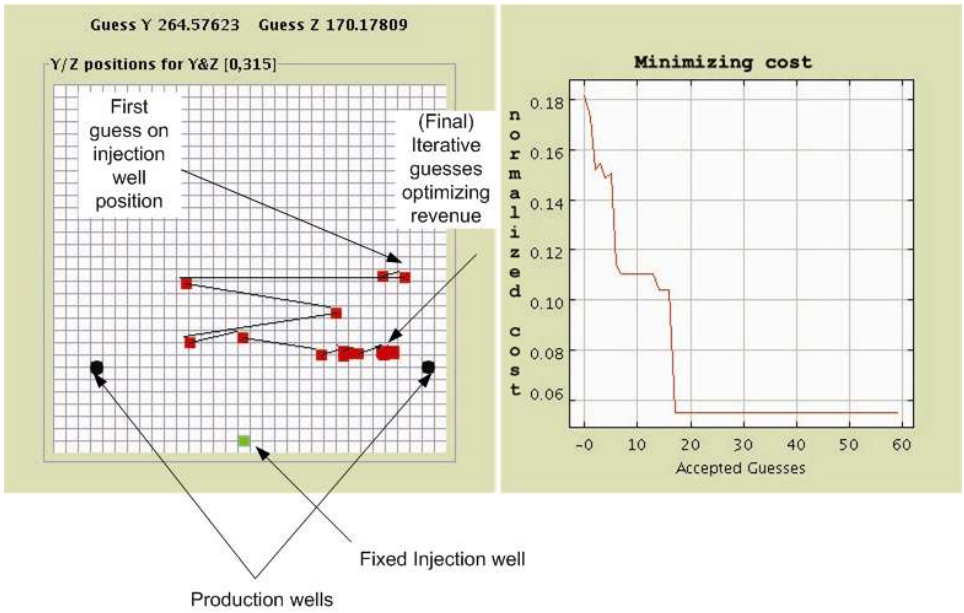


Figure 17. Well positions and normalized costs.

by the experts. The portal interface can also be used to access, monitor and steer the IPARS Factory, the VFSA Optimization service, and the Economic Model.

5.6. Sample results from the oil reservoir optimization process

Sample results from the oil reservoir optimization process are plotted in Figure 17. The plots show the well position guess produced by the VFSA optimization service and the corresponding normalized cost. The well positions plot (on the left in Figure 17) shows the oil reservoir field and the positions of the wells. The black circles are the fixed production wells. The well at the bottom most part of the plot is a fixed injection well. The plot also shows the sequence of guesses returned by the VFSA service for the other production well (shown by the lines connecting the light squares). For each guess, the plot on the right shows the corresponding normalized cost value. It can be seen that this value decreases with successive guesses until it stabilizes. This validates the optimization process.

6. SUMMARY AND CONCLUSIONS

In this paper we presented the design, development and operation of a prototype application that uses P2P interactions between applications and services on the Grid to enable the autonomic

optimization of an oil reservoir. The prototype application optimized the placement and operation of oil wells to maximize overall revenue. The application consisted of instances of distributed multi-model, multi-block reservoir simulation components provided by IPARS, simulated annealing based optimization services provided by VFSA, economic modeling services, real-time services providing current economic data (e.g. oil prices), historical data archives, and experts (scientists, engineers) connected via pervasive collaborative portals. It was built on the Pawn P2P substrate, which provided JXTA-based P2P 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.

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 P2P 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. Our current efforts at TASSL are focused on formulating programming models and developing programming and execution infrastructures to support autonomic Grid applications. Information about our research efforts is available at http://automate.rutgers.edu.

ACKNOWLEDGEMENTS

The authors acknowledge the contributions of R. Martino and J. A. Wheeler to the research presented in this paper. This research was supported in part by NSF via grant numbers ACI 9984357 (CAREERS), EIA-0103674 (NGS) and EIA-0120934 (ITR), and by DOE ASCI/ASAP (Caltech) via grant numbers PC295251 and 1052856.

REFERENCES

1. Foster I, Kesselman C. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufman: San Francisco, CA, 1999.
2. Global Grid Forum. http://www.gridforum.org [2004].
3. Tuecke S, Czajkowski K, Foster I, Frey J, Graham S, Kesselman C, Vanderbilt P, Snelling D. Grid Service Specification. http://www.gridforum.org/ogsi-wg/ [February 2004]
4. Project JXTA. http://www.jxta.org [2004].
5. Foster I, Kesselman C. Globus: A toolkit based grid architecture. *The Grid: Blueprint for a New Computing Infrastructure*, Foster I, Kesselman C (eds.). Morgan Kaufman: San Francisco, CA, 1999; 259–278.
6. Peaceman DW. *Fundamentals of Numerical Reservoir Simulation* (1st edn). Elsevier: Amsterdam, 1977.
7. Helmig R. *Multiphase Flow and Transport Processes in the Subsurface*. Springer: Berlin, 1997.
8. Russell TF, Wheeler MF. Finite element and finite difference methods for continuous flows in porous media. *The Mathematics of Reservoir Simulation*, Ewing RE (ed.). SIAM: Philadelphia, PA, 1983; 35–106.
9. Arbogast T, Wheeler MF, Yotov I. Mixed finite elements for elliptic problems with tensor coefficients as cell-centered finite differences. *SIAM Journal on Numerical Analysis* 1997; **34**(2):828–852.
10. Peszyńska M, Lu Q, Wheeler MF. Coupling different numerical algorithms for two phase fluid flow. *MAFELAP Proceedings of Mathematics of Finite Elements and Applications*, Brunel University, Uxbridge, U.K. (*Lecture Notes in Computer Science*, vol. 1845), Whiteman JR (ed.). Springer: Berlin, 1999; 205–214.
11. Chavent G, Jaffre J. *Mathematical Models and Finite Elements for Reservoir Simulation*. North-Holland: Amsterdam, 1986.
12. Sen MK, Stoffa PL. *Global Optimization Methods in Geophysical Inversion*. Elsevier: Amsterdam, 1995.

13. IPARS: Integrated Parallel Accurate Reservoir Simulator. http://www.ticam.utexas.edu/CSM/ACTI/ipars.html [2004].
14. Wang P, Yotov I, Wheeler MF, Arbogast T, Dawson CN, Parashar M, Sepehrnoori K. A new generation EOS compositional reservoir simulator. Part I: Formulation and discretization. *Proceedings of the 14th SPE Symposium on Reservoir Simulation*, Dallas, TX, June 1997. Society of Petroleum Engineers, 1997; 55–64.
15. Parashar M, Wheeler JA, Pope G, Wang K, Wang P. A new generation EOS compositional reservoir simulator. Part II: Framework and multiprocessing. *Proceedings of the 14th SPE Symposium on Reservoir Simulation*, Dallas, TX, June 1997. Society of Petroleum Engineers, 1997; 31–38.
16. Peszyńska M, Lu Q, Wheeler MF. Multiphysics coupling of codes. *Computational Methods in Water Resources*, Bentley LR, Sykes JF, Brebbia CA, Gray WG, Pinder GF (eds.). A. A. Balkema, 2000; 175–182.
17. Wheeler MF, Peszyńska M, Gai X, El-Domeiri O. Modeling subsurface flow on PC cluster. *High Performance Computing*, Tentner A (ed.). SCS, 2000; 318–323.
18. Wheeler MF, Wheeler JA, Peszyńska M. A distributed computing portal for coupling multi-physics and multiple domains in porous media. *Computational Methods in Water Resources* (*Lecture Notes in Computer Science*, vol. 1845), Bentley LR, Sykes JF, Brebbia CA, Gray WG, Pinder GF (eds.). Springer: Berlin, 2000; 167–174.
19. Lu Q, Peszyńska M, Wheeler MF. A parallel multi-block black-oil model in multi-model implementation. *SPE Journal* 2002; **7**(3):278–287.
20. Wheeler MF, Peszyńska M. Computational engineering and science methodologies for modeling and simulation of subsurface applications. *Advances in Water Resources* 2002; **25**(8–12):1147–1173.
21. Lacroix S, Vassilevski Y, Wheeler MF. Iterative solvers of the Implicit Parallel Accurate Reservoir Simulator (IPARS). *Numerical Linear Algebra with Applications* 2001; **4**:537–549.
22. Lu Q, Parashar M, Peszyńska M, Wheeler MF. Parallel implementation of multi-physics multi-block for multi-phase flow in subsurface, 2003. Submitted for publication.
23. Minkoff S, Stone CM, Bryant S, Peszyńska M, Wheeler MF. A loose coupling algorithm for fluid flow and geomechanical deformation modeling. *Journal of Petroleum Science and Engineering* 2003; **38**:37–56.
24. Mann V, Matossian V, Muralidhar R, Parashar M. Discover: An environment for Web-based interaction and steering of high-performance scientific applications. *Concurrency and Computation: Practice and Experience* 2001; **13**(8–9):737–754.
25. Parashar M, Parashar M, von Laszewski G, Verma S, Gawor J, Keahey K, Rehn N. A CORBA commodity Grid Kit. *Special Issue on Grid Computing Environments. Concurrency and Computation: Practice and Experience* 2002; **14**(13–15):1057–1074.
26. Mann V, Parashar M. Engineering an interoperable computational collaboratory on the Grid. *Special Issue on Grid Computing Environments. Concurrency and Computation: Practice and Experience* 2002; **14**(13–15):1569–1593.
27. Bhat V, Parashar M. A middleware substrate for integrating services on the Grid. *Technical Report TR-268*, Center for Advanced Information Processing, Rutgers University, November 2002.
28. Liu H, Parashar M. DIOS++: A framework for rule based autonomic applications. *Proceedings of the 9th International Euro-Par Conference (Euro-Par 2003)*, Klagenfurt, Austria (*Lecture Notes in Computer Science*, vol. 2790), Kosch H, Boszormenyi L, Hellwagner H (eds.). Springer: Berlin, 2003; 66–73.
29. Muralidhar R, Parashar M. A distributed object infrastructure for interaction and steering. *Concurrency and Computation: Practice and Experience* 2003; **15**:957–977.

