Autonomic Optimization of an Oil Reservoir using Decentralized Services *

Vincent Matossian and Manish Parashar
The Applied Software Systems Laboratory
Department of Electrical and Computer Engineering
Rutgers University, Piscataway NJ 08855, USA
{vincentm,parashar}@caip.rutgers.edu

Abstract

The Grid community is actively working on defining, deploying and standardizing protocols, mechanisms, and infrastructure to support decentralized, seamless, and secure interactions across distributed resources. Such an infrastructure will enable a new generation of autonomic applications where the application components, Grid services, resources and data interact as peers. In this paper we describe the development and operation of a prototype application that uses such peer-to-peer interactions between services on the Grid to enable the autonomic optimization of an oil reservoir.

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 [3] are investing considerable effort in developing and deploying standard protocols and services (e.g. the Open Grid Service Architecture[2]) 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 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.

For example, consider the oil production process; the 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 current weather information or economic data to optimize production or profit, and, (6) the scientists, engineers and other experts, in the field, in the laboratory and in the offices.

Furthermore, these entities need to dynamically discover and interact with one another as peers to achieve the overall application objectives. Simulation components interact with grid services to dynamically obtain necessary resources, detect current resource state, and negotiate required quality of service. The components interact with one another to accurately model the underlying physical phenomenon, and with data archives and real-time sensor data to enable history matching, dynamic data injection, and data driven adaptations (e.g. to detect bypassed oil). They may 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 revenue. Finally, the experts (scientist, engineers, and man-

*Support for this work was provided by the NSF via grants numbers ACI 9984357 (CAREERS), EIA 0103674 (NGS) and EIA-0120934 (ITR), DOE ASCI/ASAP (Caltech) via grant numbers PC295251 and 1052856.

Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03)
0-7695-1984-9/03 $17.00 © 2003 IEEE

agers) collaboratively access, monitor, interact with, and steer the simulations and data at runtime to drive the discovery process.

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 to enable the autonomic optimization of an oil reservoir. The paper has two key objectives: (1) to use this sample application to demonstrate the feasibility and benefits of such applications and (2) 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 operation of oil wells to maximize overall revenue. The application consists 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 collaborative portals. It is built on the Pawn P2P substrate, which provides JXTA-based [7] peer-to-peer messaging services, and the DISCOVER computational collaboratory, which combines Grid infrastructure services provided by Globus [2] and interaction and collaboration services.

The rest of this paper is organized as follows. Section 2 describes the application and introduces the participating components and services. Section 3 presents an overview of the Pawn interaction middleware. Section 4 describes the overall operation and integration of Pawn with the application, and presents results. Section 5 presents our conclusions.

2 Prototype Application: Autonomic Oil Reservoir Optimization

The goal of the prototype application described in this section is to dynamically and autonomically optimize the placement and configuration of oil wells to maximize revenue. The process is autonomic in the sense that the oil reservoir simulation utilizes peer services to adapt, configure, and optimize itself without human intervention. The overall application scenario is illustrated in Figure 1. The peer components involved are described below.

Integrated Parallel Accurate Reservoir Simulator (IPARS): IPARS[9] 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 execute on distributed computing systems. The simulator supports three dimensional transient flow of multiple phases containing multiple components plus immobile phases (rock and adsorbed components). Phase densities and viscosities may be arbitrary functions of pressure and composition or may be represented by simpler functions. There are currently ten *physical* models in IPARS, including multiphase gas–oil–water and air–water flow and reactive transport models. IPARS is primarily implemented in Fortran and C and is integrated using C++ wrappers and the Java Native Interface.

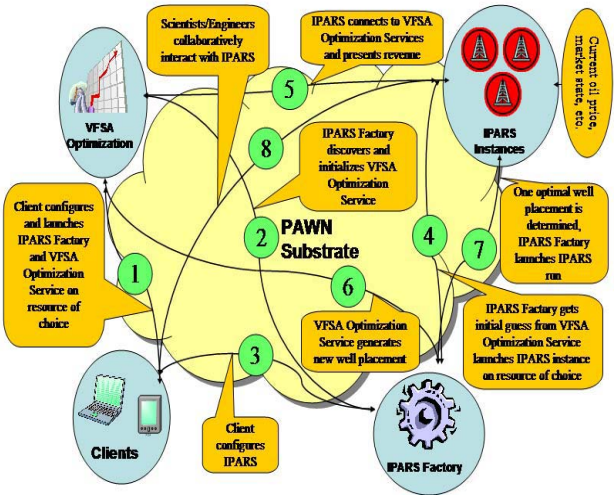

Figure 1. Oil reservoir optimization

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[4] and Globus[2].

Very Fast Simulated Annealing (VFSA): The VFSA[8] optimization service is based on statistical physics and the analogy between the model parameters of an optimization problem and particles in an idealized physical system. 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, with associated cost or energy. It then draws a new model from a flat distribution of models within the predefined limits. The associated energy 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

Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03)
0-7695-1984-9/03 $17.00 © 2003 IEEE

of the new state is larger than the initial state, the new model is accepted with a defined probability. This process is repeated for 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. Very Fast Simulated Annealing (VFSA) is a variant of SA that speeds up the annealing process. In VFSA, 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 can have its own cooling schedule and model space sampling scheme, 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[8].

Economic Modeling Service: The Economic Modeling Services use 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 general, economic value of production is a function of the time of production and of injection rates in the reservoir. It takes into account fixed costs such as drilling a well, injection, extraction, disposal, and removal of the hydrocarbons, as well as additional fluids which are used in the injection process, or are being extracted from the field.

DISCOVER Computational Collaboratory: DISCOVER[4] is an 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. Key components of the DISCOVER collaboratory include:

- **DISCOVER Interaction & Collaboration Substrate**[4] 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. The DISCOVER middleware also integrates DISCOVER collaboratory services with the Grid services provided by the Globus Toolkit [2] using the CORBA Commodity Grid (CORBA CoG) Kit [6]. Clients can now use the services provided by the CORBA CoG Kit to discover available resources on the Grid, to allocate required resources and 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)**[5] 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 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**[4] that provide the experts (scientists and engineers) with collaborative access to other peers components. Using these portals, experts can discover and allocate resources, configure and launch peers, and monitor, interact with, and steer a peer's execution process. The portal provides a replicated shared workspace architecture and integrates collaboration tools such as chat and whiteboard. It also integrates "Collaboration Streams," which maintain a navigable record of all client to client and client to application interactions and collaboration.

Using the 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 on

Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03)
0-7695-1984-9/03 $17.00 © 2003 IEEE

figure 1, 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, 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 now interacts with the Economic model to determine current revenues, and discover and interact with the VFSA service when it needs optimization (step 5). VFSA provides the IPARS Factory with optimized well information (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.

3 The Pawn Substrate

A conceptual overview of the Pawn P2P substrate is presented in Figure 2 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". Figure 3 shows the high-level architecture of the Oil Reservoir Optimization application. The application uses Pawn as a supporting framework and messaging substrate to enable the interactions between the participating entities. Pawn builds on the current Java implementation of the

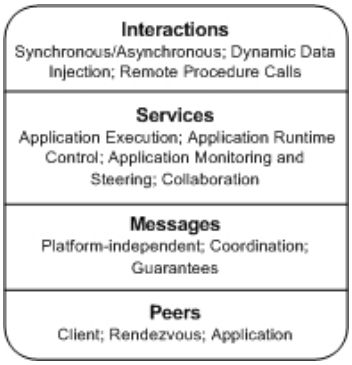

Figure 2. Conceptual architecture of Pawn.

JXTA protocols, which are summarized below.

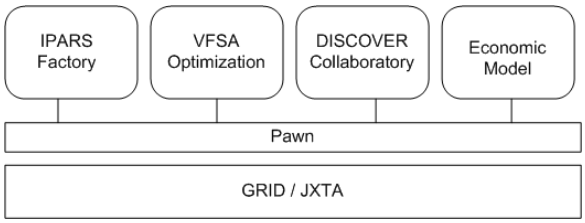

Figure 3. Overview of the oil reservoir optimization application.

3.1 Overview of Project JXTA

Project JXTA[7] is a general-purpose peer-to-peer framework introduced by SUN Microsystems in April 2001. The main drive behind JXTA is to provide developers with an open, platform and language agnostic framework for deploying interoperable peer-to-peer applications and services.

JXTA Concepts: JXTA concepts include peers, peergroups, advertisements, modules, pipes, rendezvous and security. A *peer* is any compute capable device on the network. A *peergroup* is a group of peers that share a common set of services. An *advertisement* is a language agnostic document (using XML) that describes a resource or functionality. A *module* is a piece of code (i.e. functionality) that can be initialized and started at runtime. A *pipe* is a virtual communication abstraction that provides input and output channels to a peer. A *rendezvous* peer acts as a relay as well as a distribution peer for messages and advertisements. Any peer can be a rendezvous peer, and a rendezvous can join or leave the network at any time. In JXTA, *security* is provided by the routing and transport layers, that guarantees integrity, authenticity, and confidentiality by encrypting messages before transmission.

JXTA Protocols: JXTA defines protocols for : (1) discovering peers (Peer Discovery Protocol, PDP), (2) binding virtual end-to-end communication channels between peers (Pipe Binding Protocol, PBP), (3) resolving queries and responses (Peer Resolver Protocol, PRP), (4) obtaining information on a particular peer, such as its available memory or CPU load (Peer Information Protocol, PIP) (5) propagating messages in a peergroup (Rendezvous Protocol, RVP), and (6) determining a route from a source to a destination using available transport protocols (Endpoint Routing Protocol, ERP). Protocols in JXTA define the format of the messages exchanged as well as the behavior adopted on receiving a message.

Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03)
0-7695-1984-9/03 $17.00 © 2003 IEEE

3.2 Types of Peers in Pawn

In Pawn, peers can implement one or more services (behaviors); the combination of services implemented by a peer defines its role. Typical roles for a peer are client, application or rendezvous.

A client peer deploys applications on resources and accesses them for interactive monitoring and/or steering; it also collaborates with other peers in the peergroup.An application peer exports its application interfaces and controls to the peergroup; these interfaces are used by other peers to interact with the application. The rendezvous peer distributes and relays messages, and filters them en route to their destination.

3.3 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 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.

Application Execution service [AEX]: The Application Execution service 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.

Application Monitoring and Steering service [AMS]: The Application Monitoring and Steering 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).

Application Runtime Control service [ARC]: The application runtime control 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 (i.e. PUSH and Request/Response interactions).

Collaboration Service [Group communication, Presence]: The collaboration service provides collaborative tools and support for group communication and detection of presence. 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 (Transaction and PULL interactions).

3.4 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 an entire peergroup. The pipe and resolver service use the available underlying transport protocol (TCP, HTTP, TLS). To realize 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.

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

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. 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.

Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03)
0-7695-1984-9/03 $17.00 © 2003 IEEE

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 semantics with stateful messaging and guarantee mechanism. Figure 4 presents a request/response interaction between the AMS and AEX service, 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 AEX. The receiving peer queues the message before processing it to maintain ordering of application-level messages. Once the message is processed, a similar mechanism is used to send back a response to 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.

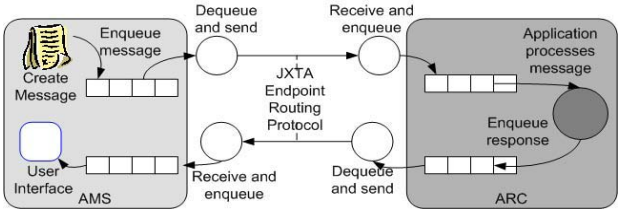

Figure 4. AMS - ARC interaction.

Dynamic Data Injection: In Pawn, every peer advertisement contains a pipe advertisement, which uniquely identifies 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 at runtime to the application/service identified by the handler tag field.

Remote Procedure Calls (PawnRPC): 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 of the methods are published as part of the peer advertisement during peer discovery. The PawnRPC XML message is a composition of the destination address, the remote method name, the arguments of the method, and the arguments associated types. Upon receiving a PawnRPC message, a peer locally checks the credentials of the sender, and if authorized, the peer invokes the appropriate method and 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 to tolerate failure.

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, 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.

Figure 5. Peer deployment.

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 with Pawn using the Java Native Interface. Figure 5 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's Meta Directory Service (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 Resource Allocation Management (GRAM) service. The portal also gives the Expert access to already deployed services and applications for collaborative monitoring and steering using DISCOVER services.

Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03)
0-7695-1984-9/03 $17.00 © 2003 IEEE

Peer Initialization and Discovery On 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 recognize 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.

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 the 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.

Oil Reservoir Optimization The reservoir optimization process consists of 2 phases, an initialization phase and an iterative optimization phase as described below.
Initialization phase: In the initialization phase, VFSA

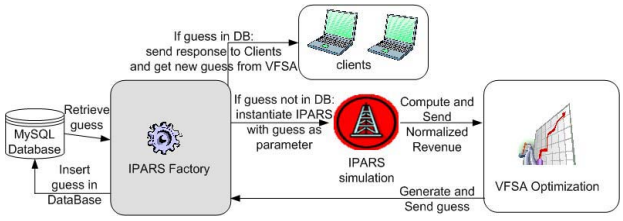

Figure 6. Optimization process.

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 periodically estimate the current revenue. This revenue is normalized and then communicated to the VFSA service, which in turn uses this value to generate an updated guess of the well parameters 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 6 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 (MySQL database) maintained by a dedicated peer. During the optimization process, when a new set of well parameters are 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 the document exchanges typically supported by P2P systems. In Pawn, these interactions are enabled by PawnRPC that provides the same semantics as the traditional RPC in a client-server system, but is implemented in a purely P2P manner.

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

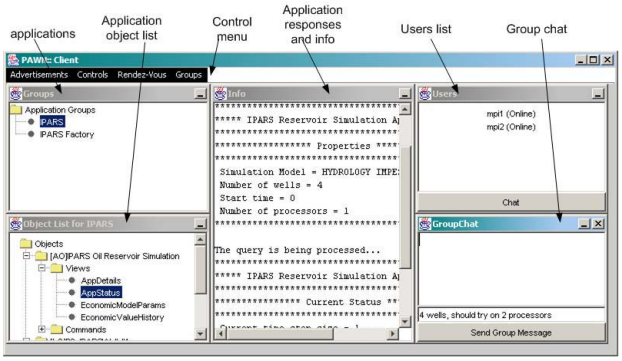

Figure 7. Expert's portal interface.

Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03)
0-7695-1984-9/03 $17.00 © 2003 IEEE

Figure 7 presents the client peer's portal interface used by the Experts.The Figure shows the Collaboration Services provided, including Chat and Whiteboard.

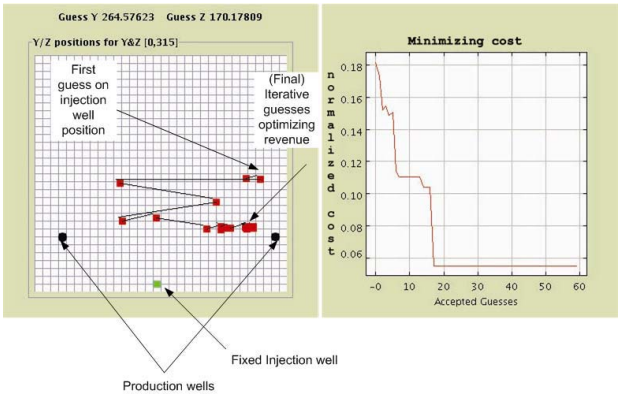

Figure 8. Sample results: well positions and normalized costs.

Sample Results from the Oil Reservoir Optimization Process Sample results from the oil reservoir optimization process are plotted in Figure8. The plots show the well position guesses produced by the VFSA optimization service and the normalized cost. The well positions plot (on the left in Figure 8) shows the oil reservoir field and the positions of the wells. The black circles are the fixed injection wells. The well at the bottom most part of the plot is a fixed production well. The plot also shows the sequence of guess 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. These results show that the optimization process required 20 iterations for this example.

5 Summary and Conclusions

This paper presented the design and implementation of the Pawn P2P substrate, and described its use in enabling the autonomic optimization of an oil reservoir using decentralized services. Pawn provides a stateful and guaranteed messaging to enable key application-level interactions such as synchronous and asynchronous communication, dynamic data injection, and remote procedure calls. The oil reservoir optimization process presented, extensively leverages these interactions to enable dynamic and seamless interactions between the participating peers to enable a global scientific investigation. The participating peers in the sample application were IPARS reservoir simulation framework, VFSA optimization service, DISCOVER computational collaboratory and Grid middleware, economic model, and client portals. Sample outputs from the optimization process were presented.

6 Acknowledgments

We would like to acknowledge the contributions of Viraj Bhat, Ryan Martino, Malgorzata Peszyńska, Mrinal Sen, Paul Stoffa and Mary Wheeler to this effort. Viraj Bhat was responsible for the CORBACoG and its integration with DISCOVER. Malgorzata Peszyńska, Mrinal Sen, Paul Stoffa and Mary Wheeler were jointly responsible with the authors for the design of the overall application scenario. Ryan Martino, Malgorzata Peszyńska and Mary Wheeler provided the IPARS framework, the economic model, and the IPARS simulation parameters. Mrinal Sen and Paul Stoffa provided the VFSA optimization service. We are specially thankful to Malgorzata Peszyńska for her help in setting up the experiments and the overall integration process.

References

[1] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, USA, 1998.

[2] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, June 2002.

[3] Global Grid Forum. Internet: http://www.gridforum.org.

[4] 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):737–754, 2001.

[5] R. Muralidhar and M. Parashar. A Distributed Object Infrastructure for Interaction and Steering. In J. G. R. Sakellariou, J. Keane and L. Freeman, editors, *Proceedings of the 7th International Euro-Par Conference (Euro-Par 2001),Lecture Notes in Computer Science*, volume 2150, pages 67–74, Manchester, UK, August 2001. Springer-Verlag.

[6] M. Parashar, G. V. Laszewski, S. Verma, J. Gawor, K. Keahey, and H. N. Rehn. A CORBA Commodity Grid Kit. *Special Issue on Grid Computing Environments, Concurrency and Computation: Practice and Experience*, 14:1057–1074, 2002.

[7] Project JXTA. Internet: http://www.jxta.org.

[8] M. K. Sen and P. L. Stoffa. *Global Optimization Methods in Geophysical Inversion*. Advances in Exploration Geophysics 4. Elsevier Science, New York, NY, USA, 1995.

[9] J. A. Wheeler and M. Peszyńska. IPARS: Integrated Parallel Reservoir Simulator. Internet: http://www.ticam.utexas.edu/CSM. Center for Subsurface Modeling, University of Texas at Austin.

Proceedings of the International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'03)
0-7695-1984-9/03 $17.00 © 2003 IEEE