Driving Scientific Applications by Data in Distributed Environments

Joel Saltz[1], Umit Catalyurek[1], Tahsin Kurc[1], Mike Gray[1], Shannon Hastings[1], Steve Langella[1], Sivaramakrishnan Narayanan[1], Ryan Martino[2], Steven Bryant[2], Malgorzata Peszynska[2], Mary Wheeler[2], Alan Sussman[3], Michael Beynon[3], Christian Hansen[3], Don Stredney[4], and Dennis Sessanna[4]

[1] Department of Biomedical Informatics, The Ohio State University
http://medicine.osu.edu/informatics
[2] Center for Subsurface Modeling, The University of Texas at Austin
http://www.ticam.utexas.edu/CSM
[3] Department of Computer Science, University of Maryland
http://www.cs.umd.edu/projects/adr
[4] Interface Laboratory, The Ohio Supercomputer Center
http://www.osc.edu

Abstract. Traditional simulation-based applications for exploring a parameter space to understand a physical phenomenon or to optimize a design are rapidly overwhelmed by data volume when large numbers of simulations of different parameters are carried out. Optimizing reservoir management through simulation-based studies, in which large numbers of realizations are sought using detailed geologic descriptions, is an example of such applications. In this paper, we describe a software architecture to facilitate large scale simulation studies, involving ensembles of long-running simulations and analysis of vast volumes of output data. This architecture is built on top of two frameworks we have developed: IPARS and DataCutter. These frameworks make it possible to implement tools and applications to run large-scale simulations, and generate and investigate terabyte-scale datasets efficiently.

1 Introduction

Numerical simulations provide a powerful mechanism for investigating and understanding complex systems and the interactions between various entities in those systems, and for effectively exploring design alternatives. In such applications, a large ensemble of simulations are carried out using different parameter values that descripe different potential initial states of the complex system under study. These applications are highly data-driven. Choosing the next set of simulations to be performed requires analysis of data from earlier simulations. As high-performance parallel and distributed platforms become more ubiquitous, traditional simulation approaches are overwhelmed by the vast volumes of data that need to be queried and analyzed. In this paper, using oil reservoir management applications as an example, we describe the design and a prototype implementation of a software system for large scale simulation studies.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 355–364, 2003.

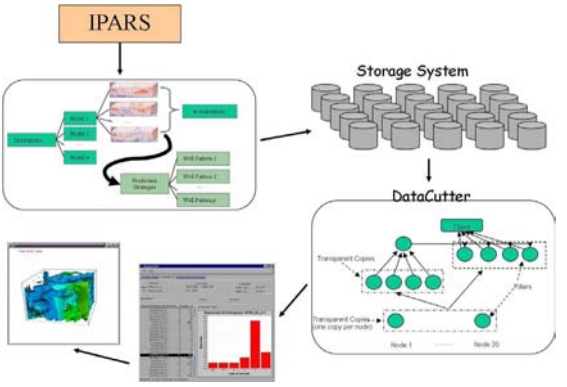


Fig. 1. Software system architecture

Numerical simulation of oil and gas reservoirs can aid the design and implementation of optimal production strategies. With a better understanding of oil and gas produced from existing reservoirs, better techniques can be devised to locate new reserves and maximize oil production from the existing reserves. Complex models of subsurface allow better assessment of the risk to the environment of existing and new reservoirs and remediation and storage of hazardous waste.

Despite technological advances in methods of determining reservoir properties, operators still have at best a partial knowledge of critical parameters such as rock permeability which govern production rates. Thus a major challenge to these objectives is incorporating geologic uncertainty while maintaining operational flexibility in large, detailed flow models. One approach to this problem is to simulate alternative production strategies (number, type, timing and location of wells) applied to multiple realizations of multiple geostatistical models. In a typical study, a scientist runs an ensemble of simulations to study the effects of varying oil reservoir properties (e.g., permeability, oil/water ratio, etc.) over a long period of time. With the help of high-performance computers, even for relatively coarse descriptions, this approach can lead to unmanageably large volumes of output data. Storage, analysis and visualization of large volumes of data generated by an ensemble of simulations is key to achieve a better understanding and characterization of oil reservoirs.

Figure 1 illustrates the overall architecture of the software system for large scale oil reservoir simulation studies. The system consists of two main frameworks that we have developed.

- **IPARS (Integrated Parallel Accurate Reservoir Simulator)** is a framework that supports multiple physical models and algorithms for the solution of multiphase flow and transport problems in porous media. The framework provides common memory management for general geometric grids, portable parallel communication, linear solvers with state-of-the-art preconditioners, keyword input, and output with visualization.

- **DataCutter** is a middleware framework for subsetting and processing multi-dimensional datasets in a distributed environment. The application processing structure is implemented as a set of interacting components, referred to as filters. Application filters can be placed on the machines in a setting so that communication and computation overheads are minimized.

In this architecture, IPARS is used to simulate alternative production strategies for a large number of geostatistical realizations of a hypothetical reservoir. Output from IPARS is stored on distributed collections of disk-based storage systems for interactive data analysis. Using the DataCutter framework, various data analysis operations can be implemented that query and manipulate those datasets. These operations can be executed on the storage systems where the datasets are stored or on other machines dispersed across a network. A graphical user interface allows a scientist to formulate queries to carry out different analysis scenarios, such as economic ranking of the alternatives, exploration of the physical basis for differences in behavior between realizations, in particular to identify regions of bypassed oil, and identification of representative realizations, which could be used as a basis for further optimization. Visualization of the datasets can be carried out remotely.

In the rest of the paper, we present the components of the software system in more detail. We describe the implementation of several data analysis scenarios for a sample case study.

2 System Components

2.1 IPARS Framework

IPARS [6, 18] represents a new approach to reservoir simulator development, emphasizing modularity of code, portability to many platforms, and ease of integration with other software. It models multiphase, multiphysics flow in porous media, and is suitable for massively parallel computers or clusters of workstations. There are currently ten physical models in IPARS, including multiphase gas–oil–water and air–water flow and reactive transport models. These models can be coupled for multiphysics simulations including couplings between IPARS models or with external codes. For example, the IPARS black–oil model was used in a loosely coupled geomechanics and flow implementation driven by reservoir subsidence problems. Solvers used by IPARS employ state-of-the-art techniques for nonlinear and linear problems including multigrid and other preconditioners.

A key feature of the IPARS framework is that it explicitly builds upon the multiblock multiphysics approach ([14, 18, 10]) which allows for the mathematically rigorous treatment of multiple domains in which different physical processes are occurring, as well as providing a basis for implementing different numerical schemes in different parts of the domain using nonmatching grids while preserving mass and momentum conservation.

The black-oil model implemented in IPARS is a three phase (water, oil and gas) model describing the flow in a petroleum reservoir [13] with three components. As such it can be considered as a subset of a *compositional* model [13].

Here it is assumed that no mass transfer occurs between the water phase and the other two phases and that water phase can be identified with water component. In the hydrocarbon (oil-gas) system, only two components: light and heavy hydrocarbons, are considered. The black-oil model described here has been shown [11] to give accurate results by comparing them with available analytical solutions and in other cases with solutions obtained by a recognized industrial reservoir simulation tool Eclipse [4]. In addition, the models and solvers under IPARS Framework were shown to be scalable in parallel [17].

2.2 DataCutter

A number of toolkits integrate processing with parallel data retrieval on tightly-coupled systems [5, 3]. Component-based frameworks provide an viable programming environment for application development in distributed environments. Besides the ease of complex application development, such models facilitate application implementations that can adapt to the heterogeneous and dynamic nature of the environment. Several research projects have focused on developing different types of component-based models [7, 12, 15].

DataCutter [2, 1] is a component framework designed to support subsetting and processing of large datasets. DataCutter implements a filter-stream programming model for developing data-intensive applications. In this model, the application processing structure is implemented as a set of components, referred to as *filters*, that exchange data through a *stream* abstraction. The interface for a *filter*, consists of three functions: (1) an initialization function (*init*), in which any required resources such as memory for data structures are allocated and initialized, (2) a processing function (*process*), in which user-defined operations are applied on data elements, and (3) a finalization function (*finalize*), in which the resources allocated in *init* are released. Filters are connected via *logical streams*. A *stream* denotes a uni-directional data flow from one filter (i.e., the producer) to another (i.e., the consumer). A filter is required to read data from its input streams and write data to its output streams only. We define a *data buffer* as an array of data elements transferred from one filter to another. The current implementation of the logical stream delivers data in fixed size buffers, and uses TCP for point-to-point stream communication.

The overall processing structure of an application is realized by a *filter group*, which is a set of filters connected through logical streams. When a filter group is instantiated to process an application query, the runtime system establishes TCP/IP socket connections between filters placed on different hosts before starting the execution of the application query. Filters placed on the same host execute as separate threads. An application query is handled as a *unit of work* (UOW) by the filter group.

The programming model provides several abstractions to facilitate performance optimizations. A *transparent filter copy* is a copy of a filter in a filter group. The filter copy is transparent in the sense that it shares the same *logical* input and output streams of the original filter. A transparent copy of a filter can be made if the semantics of the filter group are not affected. That is, the output

of a unit of work should be the same, regardless of the number of transparent copies. The transparent copies enable data-parallelism for execution of a single query, while multiple filter groups allow concurrency among multiple queries. The filter runtime system maintains the illusion of a single logical point-to-point stream for communication between a logical producer filter and a logical consumer filter. For distribution between transparent copies, the runtime system supports a Round-Robin (RR) mechanism and a Demand Driven (DD) mechanism based on the buffer consumption rate. DD aims to send buffers to the filter that would process them fastest. When a consumer filter starts processing of a buffer received from a producer filter, it sends an acknowledgment message to the producer filter to indicate that the buffer is being processed. A producer filter chooses the consumer filter with the minimum number of unacknowledged buffers to send a data buffer to, thus achieving a better balancing of the load.

3 A Case Study

In this section, we describe a case study we have implemented using IPARS and DataCutter. This case study involves generation of a large collection of data sets from IPARS simulations, and implementation and execution of various data exploration scenarios.

3.1 Data Generation

We have generated a large dataset consisting of 207 separate realizations using the IPARS simulation framework. The input data for this dataset is based on the industry benchmark SPE9 problem[8] and comes from a black-oil (three phase) flow problem on a grid with 9,000 cells. At each time step, the value of seventeen separate variables is output for each node in the grid. A total of 10,000 time steps are taken and the total output stored for each realization is about 6.9 GB. The total of 207 realizations were taken from among 18 geostatistical models and 4 well configurations/production scenarios. The geostatistical models are used to randomly generate permeability fields that are characterized by statistical parameters such as covariance and correlation length. The total dataset size is roughly 1.5 Terabytes[1] and was generated and stored on a storage cluster of 50 Linux nodes (PIII-650, 128MB, Switched Fast Ethernet) with a total disk storage of 9TB.

3.2 Data Exploration Scenarios

We have implemented several data exploration scenarios using the DataCutter framework. These scenarios involve user-defined queries for economic evaluation

[1] The case study that is described in this paper was demonstrated at Supercomputing 2001; a study with a larger dataset (5 Terabytes), which is distributed across three sites (San Diego Supercomputer Center, University of Maryland, and Ohio State University), was demonstrated at Supercomputing 2002. We plan to report on the performance evaluation of the latter study in a future work.

as well as technical evaluation, such as determination of representative realizations and identification of areas of bypassed oil.

Economic Evaluation In the optimization of oil field production strategies, the objective function to be maximized is the resulting economic value of a given production strategy. The value can be measured in a variety of ways. In our model we compute both the net present value (NPV) and return on investment (ROI). In the computation of the NPV for a given realization, a query integrates over time the revenue from produced oil and gas, and the expenses from water injection and production, accounting for the time value of the resources produced. This calculation is performed for a subset of the realizations chosen by the user. As all of the well production and injection data for each realization resides in a single file in a single disk, the data access pattern for this application is relatively simple, and most computation time is spent parsing the output file. The well data is also a relatively small part of the output data at each time step, so this is not a compute and data intensive computation. Presently, the operations for the economic evaluation is implemented as a single DataCutter filter, which also performs data retrieval from the storage system.

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

We implemented a set of filters that carry out the various operations required to find the bypassed oil regions. The implementation consists of three filters. **RD – Read data filter** retrieves the data of interest from disk and writes the data to its output stream. A data buffer in the output stream contains oil velocity and oil saturation values, and corresponds to a portion of the grid at a time step in a data set. **CC – Connected component filter** performs operations to find bypassed oil pockets at a time step on data buffer received from **RD**. These oil pockets are stored in a byte array, passed to the next filter in the pipeline. Each entry of the byte array denotes a grid cell and stores if the cell is bypassed oil cell or not. The **CC** filter writes the data buffer for each time step to the output stream, which connects **CC** to the **MT** filter. **MT – Merge over time filter** performs an AND operation on the data buffers received from CC, and finds the bypassed oil pockets. The result is sent to the client.

This scenario accesses the large four-dimensional (three spatial dimensions and time) datasets which are output for each realization. Each of the output variables are written to separate files, so this computation involves the subsetting

of data spread across several files. Additionally, if the simulation was run in parallel, the data for different parts of the domain could reside on separate disks or nodes.

Representative Realization Running multiple realizations with the same geostatistical model and well configurations can give an idea of the upper and lower bounds of performance for a particular production strategy. It is also of interest to find one realization for a given production scenario that best represents the average or expected behavior. A client query to find a representative realization for a given subset of realizations computes the average of the primary IPARS unknowns – oil concentration (C_o), water pressure (W_p), gas pressure (G_p) – and then finds the realization in the subset which is closest to the average in the sense that

$$\min \sum_{\text{all grid points}} \frac{|C_o - C_{o\_avg}|}{C_{o\_avg}} + \frac{|P_w - P_{w\_avg}|}{P_{w\_avg}} + \frac{|P_g - P_{g\_avg}|}{P_{g\_avg}}$$

is realized.

The DataCutter implementation consists of four filters. **RD - Read filter** retrieves the data of interest from disk. The read filter sends data from each dataset to the **SUM** and **DIFF** filters. A data buffer from the read filter is a portion of the grid at one time step. **SUM - Sum filter** computes the sum for C_o, W_p, and G_p variables at each grid point across the datasets selected by the user. **AVG - Average filter** calculates the average for C_o, W_p, and G_p values. **DIFF - Difference filter** finds the sum of the differences between the grid values and the average values for each dataset. It sends the difference to the client, which keeps track of differences for each time step, carries out average over all time steps for each dataset.

3.3 Visualization

We have developed two different implementations for visualizing output from a realization. The first visualization employs isosurface rendering implemented using the Active Data Repository (ADR) framework [3], which is designed to support processing of large, out-of-core datasets via generalized reduction operations on distributed memory systems. The ADR implementation uses the marching cubes and polygon rendering functions of the Visualization Toolkit (VTK) [16] for extracting and rendering an iso-surface from large, out-of-core datasets on a distributed-memory parallel system with a local disk farm [9]. Figure 2 shows a visualization of bypassed oil regions using the isosurface rendering. The second visualization tool is based on direct volume rendering. We implemented a DataCutter filter using the volume rendering library, developed at Ohio Supercomputer Center. This implementation uses a texture based volume rendering approach and takes advantage of 3D hardware texture rendering (e.g., NVIDIA GeForce 3) cards.

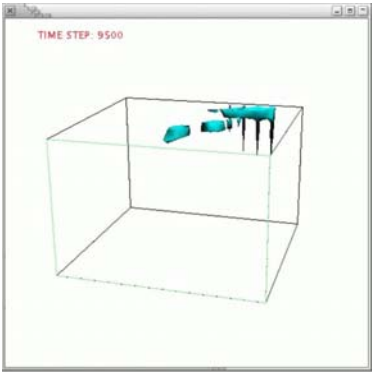
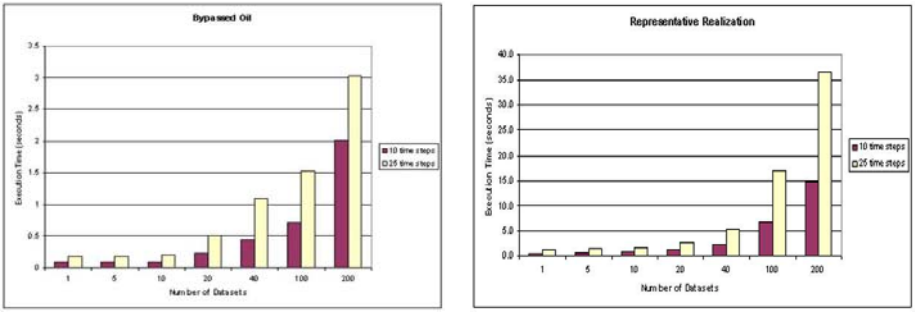


Fig. 2. Visualization of bypassed oil

4 Results

We present results for the representative realization and bypassed oil scenarios. The experiments were carried out using the DataCutter implementations of the two scenarios and the 1.5TB data set generated in this study. The data set is stored on 19 nodes of the 9.5TB storage cluster at University of Maryland. In the experiments, we varied the number of data sets accessed by a query. For this purpose, we submitted a total of 28 queries (varying the number of datasets from 1 to 200); 14 queries for the bypassed oil analysis and 14 queries for the representative realization analysis. Half of the queries in each set requests data over 10 time steps (time steps 0 through 9999, with increments of 1000 time steps), while the other half retrieves 25 time steps (time steps 0 through 9999 with increments of 400 time steps).

Figures 3 shows the execution time for each query. As is seen from the figures, up to 40 datasets the query execution time remains below 1 seconds for the bypassed oil scenario. As the number of datasets is increased the query execution time increases, as expected. For a query that accesses 200 datasets over 25 time steps, the execution time is about 3 seconds. Thus, we are able to achieve interactive rates even for queries that access a large number of datasets from the collection. The experimental results show that queries for representative realization scenario take longer, as the operations involved are more expensive. As is seen from the figure, the query execution time remains below 5 seconds for queries that access up to 40 datasets over 25 time steps. Our preliminary results show that the query execution does not scale well after 40 datasets for the representative realization scenario. This is because of the fact that in the experiments the number of transparent copies for the SUM and DIFF filters are fixed at four. In future work, we plan to examine the effect on performance of varying the number of transparent copies.



(a) (b)

Fig. 3. (a) Performance results for bypassed oil computation (b) Performance results for representative realization computation

5 Conclusions

We have demonstrated a new paradigm for applying reservoir simulation to the challenges of reservoir management. The selected challenge was to enable the evaluation of large numbers of realizations, both of geological models and of well patterns. The black oil model within the IPARS framework provided the numerical solutions to the forward flow problems, while the DataCutter middleware provided the means for subsetting and filtering the multidimensional output. The volume of data resulting from such studies can be extremely large. Such datasets would be unmanageable for most evaluation tools, especially for complex queries such as identifying representative realizations or locating regions of bypassed oil. The IPARS/DataCutter applications enable the creation, interrogation and visualization of such datasets while maintaining the familiarity and speed of interaction of the traditional simulation workflow. Thus many more realizations of higher resolution geologic models and more production strategies can be studied in greater detail within a given time, increasing the utility of the study for decision making.

Acknowledgments. The work described in this paper was partly supported under the grant of National Partnership for Advanced Computational Infrastructure (NPACI) 10181410. In additional, the first four authors were supported partially by grants DOE: DE-FG03-99ER25371, DOD: PET2/ UTA01-347, and the NSF grants: SBR 9873326, ITR EIA-0121523. This research was also supported in part by the National Science Foundation under Grants #ACI-9619020 (UC Subcontract #10152408), #EIA-0121177, #EIA-0203846, #ACI-0130437, and #ACI-9982087, Lawrence Livermore National Laboratory under Grant #B500288 and #B517095 (UC Subcontract #10184497), and the Department of Defense, Advanced Research Projects Agency, USAF, AFMC through Science Applications International Corporation under Grant #F30602-00-C-0009 (SAIC Subcontract #4400025559).

References

1. M. Beynon, C. Chang, U. Catalyurek, T. Kurc, A. Sussman, H. Andrade, R. Ferreira, and J. Saltz. Processing large-scale multidimensional data in parallel and distributed environments. *Parallel Computing*, 2002.

2. M. D. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J. Saltz. Distributed processing of very large datasets with DataCutter. *Parallel Computing*, 27(11):1457-1478, Oct. 2001.
3. C. Chang, T. Kurc, A. Sussman, and J. Saltz. Optimizing retrieval and processing of multi-dimensional scientific datasets. In *Proceedings of the Third Merged IPPS/SPDP (14th International Parallel Parallel and Distributed Processing Symposium)*. IEEE Computer Society Press, May 2000.
4. Eclipse-100, Technical Description. Schlumberger Technology Corporation. 1998.
5. S. Goil and A. Choudhary. PARSIMONY: An infrastructure for parallel multidimensional analysis and data mining. *Journal of Parallel and Distributed Computing*, 61(3):285-321, March 2001.
6. *IPARS: Integrated Parallel Accurate Reservoir Simulator.* http://www.ticam.utexas.edu/CSM/ACTI/ipars.html.
7. C. Isert and K. Schwan. ACDS: Adapting computational data streams for high performance. In *14th International Parallel & Distributed Processing Symposium (IPDPS 2000)*, pages 641-646, May 2000. IEEE Computer Society Press.
8. J. E. Killough. Ninth (SPE) comparative solution projection: a reexamination of black-oil simulation. The 13th Symposium on Reservoir Simulation, pages 135-147, February 1995. SPE 29110.
9. T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J. Saltz. Visualization of large datasets with the Active Data Repository. *IEEE Computer Graphics and Applications*, 21(4):24-33, July/August 2001.
10. Q. Lu, M. Peszynska, and M. F. Wheeler. A parallel multi-block black-oil model in multi-model implementation. In *2001 SPE Reservoir Simulation Symposium*, Houston, Texas, 2001. SPE 66359.
11. Q. Lu. *A Parallel Multi-Block / Multi-Physics Approach for Multi-Phase Flow in Porous Media.* PhD thesis, University of Texas at Austin, Austin, Texas, 2000.
12. R. Oldfield and D. Kotz. Armada: A parallel file system for computational. In *Proceedings of CCGrid2001: IEEE International Symposium on Cluster Computing and the Grid*, May 2001. IEEE Computer Society Press.
13. D. W. Peaceman. *Fundamentals of numerical reservoir simulation.* Elsevier Scientfic Publishing Company, Amsterdam-Oxford-New York, first edition, 1977.
14. M. Peszynska. Advanced techniques and algorithms for reservoir simulation III. Multiphysics coupling for two phase flow in degenerate conditions. In John Chadam, Al Cunningham, Richard E. Ewing, Peter Ortoleva, , and Mary Fanett Wheeler, editors, *IMA Volumes in Mathematics and its Applications, Volume 131: Resource Recovery, Confinement, and Remediation of Environmental Hazards*, pages 21-40. Springer, 2002.
15. B. Plale and K. Schwan. dQUOB: Managing large data flows using dynamic embedded queries. In *IEEE International High Performance Distributed Computing (HPDC)*, August 2000.
16. W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach To 3D Graphics.* Prentice Hall, 2nd edition, 1997.
17. M. F. Wheeler, M. Peszynska, X. Gai, and O. El-Domeiri. Modeling subsurface flow on PC cluster. In A. Tentner, editor, *High Performance Computing*, pages 318-323. SCS, 2000.
18. M. F. Wheeler, J. A. Wheeler, and M. Peszynska. A distributed computing portal for coupling multi-physics and multiple domains in porous media. In L. R. Bentley, J. F. Sykes, C. A. Brebbia, W. G. Gray, and G. F. Pinder, editors, *Computational Methods in Water Resources*, pages 167-174. A. A. Balkema, 2000.