Modelling Data-driven CO_2 Sequestration Using Distributed HPC Cyberinfrastructure

Yaakoub el-Khamra
Craft and Hawkins Dept. Of Petroleum Engineering
Louisiana State University
Texas Advanced Computing Center
The University of Austin Texas
yaakoub@tacc.utexas.edu

Shantenu Jha
Center for Computation and Technology
Louisiana State University
Department of Computer Science
Louisiana State University
e-Science Institute, Edinburgh, UK
sjha@cct.lsu.edu

Christopher D. White
Craft and Hawkins Dept. Of Petroleum Engineering
Louisiana State University
Center for Computation and Technology
Louisiana State University
cdwhite@lsu.edu

ABSTRACT

In this paper, we layout the computational challenges involved in effectively simulating complex phenomenon such as sequestering CO_2 in oil and gas reservoirs. The challenges arise at multiple levels: (i) the computational complexity of simulating the fundamental processes; (ii) the resource requirements of the computationally demanding simulations; (iii) the need for integrating real-time data (intensive) and computationally intensive simulations; (iv) and the need to implement all of these in a robust, scalable and extensible approach. We will outline the architecture and implementation of the solution we develop in response to these requirements, and discuss results to validate claims that our solution scales to effectively solve desired problems sizes and thus provides the capability to generate novel scientific insight.

Categories and Subject Descriptors

D.1.3 [**Concurrent Programming**]: Parallel Programming

General Terms

ALGORITHMS,DESIGN,PERFORMANCE,MANAGEMENT

Keywords

Ensemble Simulations, Sequestration, Distributed Applications, Scale-out

1. INTRODUCTION AND MOTIVATION

Global energy needs today present serious challenges: the increasing demand for energy must be met, however at the same time the emissions of greenhouse gases into the atmosphere must be reduced. Even as alternative energy sources continue to develop and gain popularity, the fact remains that fossil carbon resources will continue to be in heavy use (in both developing and industrialized countries) and consequently generate large volumes of carbon dioxide [4]. The atmospheric impact of this greenhouse gas can be abated through capturing and sequestering significant fractions of the produced CO_2.

For long-term storage of large volumes of CO_2, porous subsurface geologic formations are ideal candidates: these are the same formations responsible for the existence of oil and gas reservoirs. Indeed much of the technology behind carbon dioxide sequestration (including drilling, gas injection, reservoir management and of course reservoir simulation) stems from drilling, petroleum and reservoir engineering. Injecting CO_2 into an oil-gas reservoir can also lead to improved oil recovery by "pushing out" the oil and gas for production, this allows for reduced net cost through increased revenues from oil and gas production [5].

One of the major areas of research in this field is the characterization of reservoirs that are safe and secure, environmentally and geologically and are therefore promising candidates for CO_2 sequestration [4, 20]. Our efforts are directed towards developing cyberinfrastructure tools, technologies and abstractions that facilitate large scale reservoir characterization and forecasting studies. Since the amount of information obtained directly from reservoirs is very small compared to the actual size of the reservoir, history matching techniques have been developed to match actual reservoir production with simulated reservoir production, therefore obtaining a more "satisfactory" set of models. One of the promising approaches to history matching is the use of Ensemble Kalman filters (EnKF) [15, 11, 17, 10].

This paper is structured as follows: in the next section we describe the basic physical processes and parameters that are used to understand sequestration. In the latter part of Section 2, we outline the computational approach adopted, computational complexity and the simulation parameter choice considerations that need to be determined. In Section 3 we present the architecture and the specific solution we implement to address the physical and engineering problem of CO_2 sequestration. The focus in this section is not on the specific details of each component – as each component of the solution was developed for a different original problem(s), but on how we bring the various components together. The final section outlines some preliminary results that validate our solution – both computational and scientific (inject rates for CO_2).

2. MODELLING CO_2 SEQUESTRATION

2.1 The Computational Approach

CO_2 sequestration in oil/gas reservoirs typically involves history matching studies to obtain a better understanding of the reservoir geology and fluid flow properties. At the core of the history matching process is the EnKF – which are recursive filters that can be used to handle large, noisy data; the data in this case would be the results and parameters from an ensemble of reservoir models that are sent through the filter to obtain the "true " state of the reservoir. The models start with permeability, porosity (possibly pressure and saturation) field maps that are generated using geostatistically methods from a base case. If we start with a close enough base case, generate an ensemble of representative models and run sufficient analysis steps, the true state of the reservoir will lie somewhere between statistically representative ensemble members.

Since the reservoir model varies from one ensemble to another, the run-time characteristics of the ensemble simulation are irregular and hard to predict. Furthermore, at simulation times when real historical data is available, all the data from the different ensembles at that simulation time must be compared to the actual production data (i.e. analysis step), before the simulations are allowed to proceed. This translates into a global synchronization point for all ensembles; hence performing large scale studies for complex reservoirs in a reasonable amount of time would benefit greatly from the use of distributed, high performance, high throughput and on-demand computing resources.

These history matching/reservoir characterization studies are then followed by forecast studies where CO_2 is injected at specified rates and pressures based on operational and regulatory constraints. These simulations answer key questions about the sequestration process: where will the CO_2 gas flow, how much CO_2 can be injected safely, what is the long term fate of the injected CO_2.

For our purposes we make the same simplifying assumptions about the injected CO_2, as in Ref [21]. Carbon dioxide is treated as a gas phase that is soluble in oil but not water, and compositional effects of CO_2 injection are ignored. In our test/demonstration simulations, carbon dioxide is injected as a supercritical gas at 2800 psi. Regulatory constraints limit the bottom-hole injection pressure; the bottom hole pressure at the injection point does not exceed the hydrostatic pressure gradient by more than 0.2 psi/ft.

2.2 Analyzing The Computational Complexity

An important parameter in the EnKF workflow is the size of the ensemble (number of ensemble members). The ensemble size should be large enough to propagate the information contained in the observations to the model variables. Smaller ensembles cause the analysis error to become larger. Too small an ensemble size can give poor approximations to the infinite ensemble case: the ensemble covariance underestimates the error covariance substantially. This effect can be monitored by comparing the actual forecasted model data differences to those expected on the basis of the forecasted ensemble covariance [9].

In contrast, Hout et al [13] found that while the analysis error decreased as the number of ensembles increased, i.e. representative ensembles can be maintained with a relatively small ensemble size (order of hundreds) using a pair of ensembles. Furthermore, the required ensemble size remains constant irrespective of the state vector size and observation size [12]. However, if a small ensemble size requiring severe localization (as is sometimes the case with reservoir models), this can cause a significant amount of spatial incoherence.

For these reasons, optimal ensemble size is considerably difficult to determine without any preliminary analysis. Large ensemble sizes directly translate to increased computational cost and delays in the time to completion. Conversely, smaller ensemble sizes, while more efficient, require prior knowledge of localization requirement for reduced analysis errors.

2.2.1 *Analyzing the Trade-offs*

Typically, reservoir characterization studies followed by forecast studies are run for a reservoir management recommendation to be made. The studies must be therefore completed within a fixed time-frame (few weeks at best). This imposes restrictions on the size of the ensemble used. Fortunately, the ensemble members are independent therefore can be run across a distributed environment. It is for this reason that we develop a scalable autonomic framework for EnKF applications (to be described in Section 3). The larger ensemble sizes also increase the computational cost of the EnKF application. For that reason, we developed a parallel EnKF.

Since the choice of ensemble size has a direct impact on total time to completion, a tradeoff can be established: large ensembles of small/less detailed simulations as opposed to small ensembles of large/detailed simulations. If the ensemble is too small, it fails to span the space of possible models and the inversion tends to be unstable. On the other hand, large ensembles are inefficient. The tradeoff can keep the number of SU's [1] required constant, and with prior knowledge of available resources, the total time to completion as well.

The question becomes: given a small time-frame of ten days for example, what ensemble size and what problem size can be completed so that by the tenth day results can be collected and a decision made. While we have not addressed this problem in the workflow manager or the reservoir simulator, we have built those tools with these tradeoffs in mind.

2.2.2 *Computational Requirement: Current Capability*

Consider a practical reservoir study that involves characterization, production forecast and sequestration forecast. For a one million grid cell model with our reservoir simulator, one iteration on 16 cores requires roughly 30 seconds. Also assume we will be performing data assimilation with observational data collected every 300 iterations. Therefore each model takes 9000 seconds on 16 cores. With 100 ensemble members, this equates to 900,000 seconds using 16 cores running one ensemble at a time, or 9000 seconds on a job of 1600 cores with all 100 ensembles running concurrently (the details of which will be presented later)

Assume for now that we are working with the Ranger supercomputing system at TACC. We would have access to 1600 cores for a maximum of 2 days, which should allow us to run roughly 1.6 years worth of simulations. (1 iteration which takes 30 seconds, advances the simulated time by 0.1 day; 2 days of wall-clock time thus enables 5760 iterations, which correspond to 576 days which is $\approx$ 1.6 years). With 15 years of history matching, 15 years of forecasts and another 15 years of sequestration simulation, or 29 simulations with 1600 core size and 2 day duration. The wait time

[1] An SU is a system unit: an accounting unit amounting to one core on one node for one hour

in the queue for such a job is, on average, 30 hours (1.25 days) on Ranger. If we submit the subsequent job to the queuing system after the current job is finished, the total duration of the entire study will therefore be roughly 95 days (29 * (2 + 1.25)).

Certain improvements are possible obviously. Using three machines as a distributed infrastructure instead of just one machine, based on figure 5, we can expect a reduction of the total time to completion by about a third, roughly 65 days. Clearly this is a very long time to solution and further optimizations are both required and possible.

3. MEETING THE REQUIREMENTS: DESIGN AND ARCHITECTURE

Due to the complexity of the problem, involving reservoir simulation, geostatistics on one end, and simulation management, workflows, grid and high-throughput computing on another, a design decision was made early in development to make full use of existing tools and abstractions whenever possible. This resulted in a rapid development cycle and faster turn-over. The components we developed and use are as follows:

3.1 The Reservoir Simulator

Based on the Cactus Code [3] high performance scientific computing framework and the Portable Extensible Toolkit for Scientific Computing: PETSc [2], the BlackOil reservoir simulator solves the equations for multiphase fluid flow through porous media, allowing us to simulate the movement of oil and gas in subsurface formations. With a few parameter changes, BlackOil is also used for modelling the flow of CO_2, however it cannot simulate any geochemical interaction. While adequate as a first order approximation, it is still under heavy development to enable it to satisfactorily model geochemical phenomena.

3.2 The Ensemble Kalman filter

The Kalman filter and its variants are being widely adopted to address the need for flexible, efficient inversion of nonlinear, stochastic problems [8]. However, many challenges remain, including a better understanding of filter behavior, methods to stabilize filters, and guidelines for optimal filter structure. In addition, because ensembles of large, nonlinear models impose a heavy computing burden, engineers and scientists will need greater access to supercomputing resources. Efficient use of these resources requires a balanced cyber-infrastructure for ensemble inversion; the inversion infrastructure must (1) be sensor–aware, so that inversion proceeds when new data are available; (2) be able to manage parallel and distributed workflow elements; (3) distribute and load–balance simulations with a wide range of execution times; (4) move large data sets to initialize models, compute gains, and update models; and (5) recover from errors in ensemble member simulations.

The EnKF we use is based on Cactus and PETSc –it computes the Kalman gain matrix and updates the model parameters of the ensembles. The Kalman filter requires live production data from the reservoir for it to update the reservoir models in real-time, and launch the subsequent long-term forecast, enhanced oil recovery and CO_2 sequestration studies

3.3 Developing Scalable and Extensible Application Frameworks: A SAGA-based autonomic computing framework for scientific applications

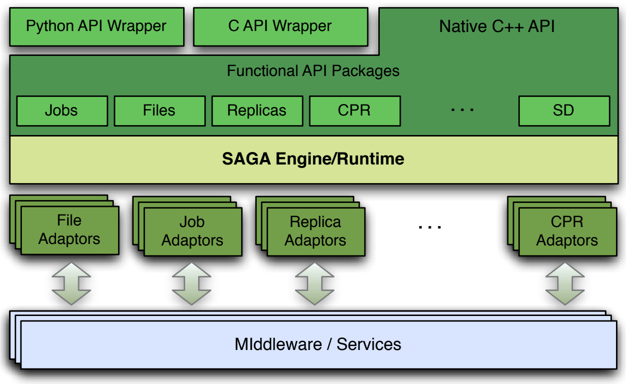


Figure 1: Layered schematic of the different components of the SAGA landscape. Middleware specific adaptors applications developed using SAGA make applications developed using SAGA grid portable. SAGA facilitates the creation of a framework made up of "abstractions" (Bigjob; ManyJob) that supports the functionality desired for distributed applications, but not confined to a specific infrastructure.

A fundamental question at the heart of all scientific applications, is how can scientific applications be developed so as to use as broad a range of infrastructure as possible, without vendor or middleware lock-in, yet with the flexibility and performance that scientific applications demand. The SAGA programming system has been developed to address precisely this question. It consists of a high-level application-oriented API, the SAGA *Engine* (a dynamic library implementing the functionality of the API), and backend, system-specific adaptors. Our recent work using SAGA [1] has overcome a traditional limitation of RE [2] simulations, viz., restricted set of platforms have been used and consequently, only small physical systems have been investigated so far. We have developed a general-purpose framework (Lazarus) that can overcome traditional limitations and can be used over a wide-range of production distributed environments [19]. Interestingly, the same framework can be used to marshall different types and sizes of resources [1]. For example, the BigJob abstraction that we first employed on 512-node Linux Clusters on LONI can also be used for monolithic usage on the 64,000 core mammoth Ranger(TACC) machine [22]. It is also important to mention that the SAGA-based framework discussed is extensible, in that it can be used to implement many other applications in addition to those discussed here.

The power to do so arises from simple design decisions: the use of appropriate programmatic and system abstractions that allow users to do what they can do best (i.e., provide the simulation and orchestration logic), whilst ensuring that the middleware used provides required services (such as checkpoint management, application monitoring and recovery) seamlessly and effectively from the application developers perspective.

As mentioned earlier, ensembles must be synchronized at each assimilation step. This makes the inversion process as slow as the slowest member. This might be addressed using (a) adaptive resource allocation and intelligent queuing; (b) use of subensembles, possibly adaptively; (c) adapting ensemble size. Thus there is a distinct advantage of utilizing distributed resources. It is also useful to note that for the end-user, the complexity of using one resource is

[2]RE: Replica Exchange

the same as using multiple resources; however, the gains increase with every added resource.

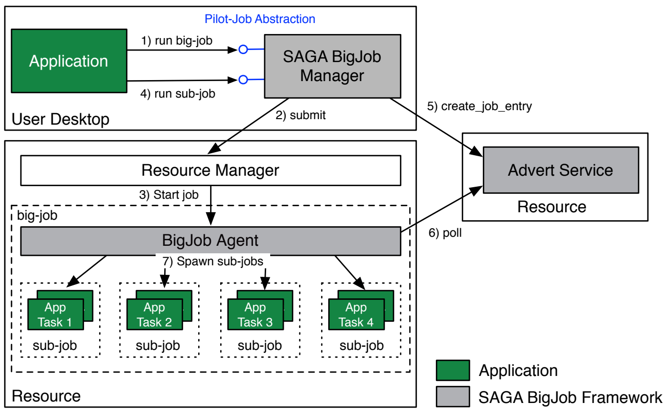


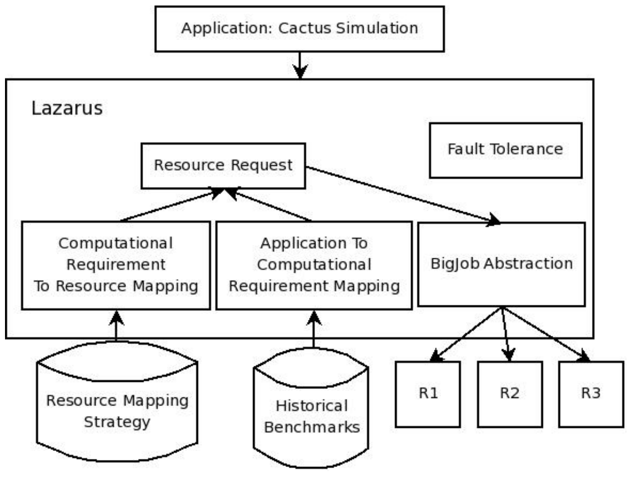
Figure 2: BigJob Architecture: The core of the framework, the BigJob Manager, orchestrates a set of sub-jobs via a BigJob Agent using the SAGA job and file APIs. The BigJob Agent is responsible for managing and monitoring sub-jobs.

Lazarus uses the BigJob abstraction to launch simulations on distributed high performance, high throughput and grid computing resources [16], the SAGA file adaptor to move files from one machine to another and BQP[3] data from a small python wrapper around the BQP command line tool to retrieve the optimal location and number of BigJobs required to satisfy the computation power required. The Lazarus resource management function queries BQP for the "optimal" job size and duration for any given resource.

For runs distributed across several machines, Lazarus needs to gather the data from all different machines into one location where the EnKF can run. To that end, the SAGA file adaptors are used to copy files over to the resource where the EnKF will run, and copied back to all other resources after the analysis stage. Since we are not working with a large scale production history matching run, the amount of data transfer involved does not impose any noticeable performance issues. Naturally, investigating the optimal location for placing the EnKF application, based upon network performance and real-time performance as discussed in Ref. [14], will find its way into the next iteration of Lazarus implementation. Lazarus also uses a user specified check for the integrity of the simulation data. In our case, this is done through various system and HDF5 calls to make sure the files exist, are not sized zero, and a user specified command using HDF5 tools (h5ls) returns successfully.

The Lazarus framework contains several aspects of autonomy: it has self-configuration (deployment on resources), self optimization (using BQP data), self monitoring (checking its own output) and of course self healing (resubmission of faulty simulations). These aspects have been implemented with varying levels of intelligence: the self optimization for example is a basic algorithm that uses BQP data to assign big-jobs to resources, but does not take into consideration bandwidth requirement and the cost of copying files across machines. The self-healing on another hand can typically resurrect jobs that fail due to node failure as opposed to software failure. Many autonomic features of Lazarus will be improved to incorporate sophisticated inherent intelligence. Lazarus is distinct to other well established and successful approaches such as Accord [18] for autonomic computing, in that it is not a component-based programming system but it is a framework already composed of multiple components. Importantly the programming system that is used is SAGA – both for the application and the Lazarus framework.

[3] BQP: Batch Queue Prediction, a tool used to predict the time spent by a job (of known size and duration) in the queuing system for a particular machine.



Application: Cactus Simulation
Lazarus
Resource Request
Fault Tolerance
Computational Requirement To Resource Mapping
Application To Computational Requirement Mapping
BigJob Abstraction
Resource Mapping Strategy
Historical Benchmarks
R1
R2
R3

Figure 3: Architecture of Lazarus – a Framework for Developing Autonomic Computational Science Applications. The "Resource Mapping" Strategy input is provided by BQP. Given the size of the individual tasks, the "Historical Benchmarks" help determine the size of the sub-jobs. (For the specific workload chosen, it so happens that the size of each sub-job is 16).

3.4 Sensors-to-Simulations (S2S)

To perform realtime reservoir characterization and forecast, live sensor data must be retrieved from the field and used in the data assimilation stage. The availability of new data also triggers a new EnKF stage in the workflow. To that end we implemented a communication layer based on data streaming from the data acquisition (sensor) platforms to a data spool in the form of a MySQL database 8. The EnKF application queries the data spool for the latest available sensor platform (field observations) data. Once the new field observations are retrieved they are labeled as "assimilated" to differentiate them from the next–in–line observations. Developed, initially using Cactus and database interfaces, to send measurement-while-drilling (MWD) data from drilling bottom-hole-assemblies to drill-string simulators, the S2S framework has been extended to accommodate sending sensor data to the BlackOil reservoir simulator (production data required for well modelling) and the Ensemble Kalman filter for history matching. The S2S framework also announces the arrival of new history data to Lazarus to trigger the launching of a new stage of ensemble simulations [7, 6]. It is worth mentioning that while currently we use a MySQL database, a more consistent solution would be to use the SAGA advert service, which would remove the need to access MySQL in Lazarus.

Putting the Components together: The use of an intermediate data–spool is necessary for many reasons: it is difficult to guarantee resource availability and synchronization with the EnKF. It is also prohibitively costly to maintain an active EnKF session throughout the life–time of the reservoir. With a data–spool we can also adjust the update rate at the EnKF session without losing any data from the reservoir. Updates to the models can be as immediate as the moment data is broadcast from the sensors in the field, or as late as when the reservoir is being abandoned. With immediate model updates, we can run forecast studies.

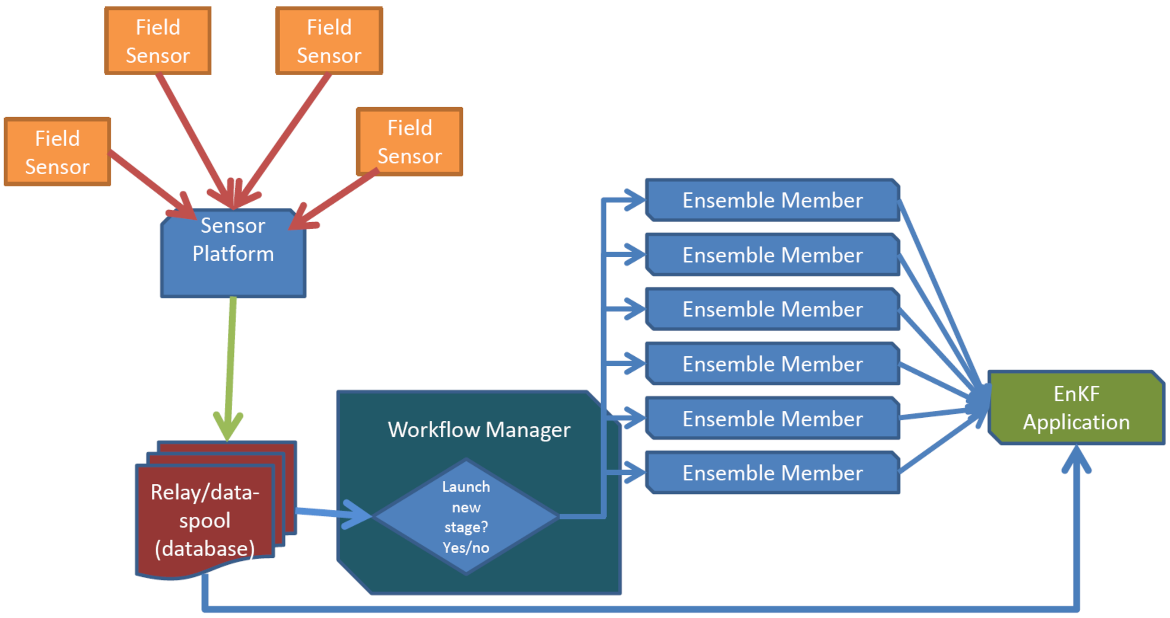


Figure 4: Schematic illustrating the interaction of BlackOil, EnKF, S2S and Lazarus. The blue arrows represent the flow of simulation data while the red arrows represent the flow of sensor data.

When used in conjunction, these components allow us to perform real–time characterization studies of large scale reservoirs as well as long term forecast, enhanced oil recovery and CO_2 sequestration studies. As more reservoir history data is generated during enhanced oil recovery and CO_2 injection, the closed-loop can continue to run and provide insight as the reservoir geological models are being updated by the EnKF. Continuously running the latest models in forecast simulations with the latest sensor data will hopefully allow us to predict un-intended movement of CO_2, leaks into aquifers, critical pressure build-ups or huge pressure drops (fractures, blow-outs etc.). To that end we use the BlackOil reservoir simulator as an early prototype of what will eventually be an accurate simulator that can model CO_2 sequestration, geochemical and geomechanical phenomena. The S2S framework is also under heavy development to add sensor metadata allowing Lazarus, through autonomic logic, to distinguish between critical and non-critical sensor data, and ultimately making use of on-demand computing resources (should the sensor data prove critical).

4. PERFORMANCE DATA AND ANALYSIS:

We have previously run an EnKF based application on several TeraGrid resources and demonstrated effective *Scale-out* across three machines [16]. Additionally, we incorporate the use of Condor pools in Lazarus for even higher throughput, demonstrating possible advantages to using multiple pilot job mechanisms.

4.1 Performance Data

Figure 5 shows the time to completion T_c for various TeraGrid machines and combinations thereof.There are several components to T_c worth discussing. The first, is the time that it takes to submit to the queuing system and time for file-transfers (in and out). In a way this is overhead, and thus we label this as $t_{overhead}$. The second component is the time that the submitted jobs wait in queue for resources requested to become available. We label this as t_{wait};

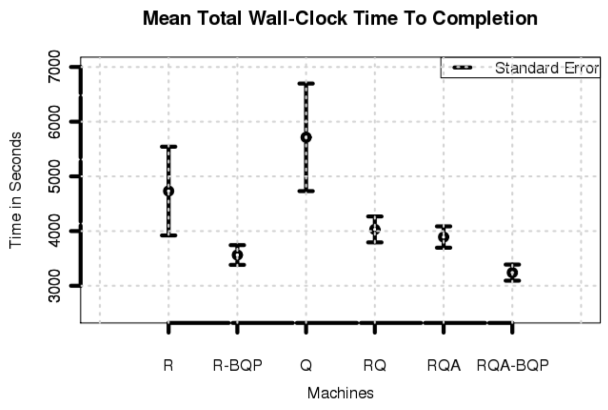


Figure 5: Time to completion T_c with Lazarus. From Left to Right: (i) Ranger (ii) Ranger when using BQP, (iii) QueenBee, (iii) Ranger and QueenBee, (iv) Ranger, QueenBee and Abe, (v) Ranger, QueenBee and Abe when using BQP. Ranger, QueenBee and Abe are TeraGrid supercomputer systems

Sample #	Machine	Queue	Num. Cores	Duration (hrs)
1-10	Ranger	normal	64	2:00
1-10	Ranger-BQP	development	128	1:00

Table 1: Table showing the configurations chosen on Ranger, with and without BQP. Notice how the use of BQP has a small, but significant change in the queue, size and duration requested.

the final component is the run-time that simulations actually take to complete, which we label as t_{run}. Thus, $T_c = t_{overhead} + t_{wait} + t_{run}$.

The experiments are performed with and without BQP for single and combined resources, starting with single machine runs. The experiments are run ten times, randomly interspersed over a period of few weeks. This is done for three different machines on the TG – Ranger, QueenBee and Abe. Ranger is the flagship machine of the Texas Advanced Computing Centre; QueenBee (QB) the flagship machine of LONI and Abe a large machine at NCSA. As the results from the single-resource configuration show, the individual resources have somewhat different intrinsic performance. We find that there are fluctuations in both the wait-time in queue and consequently the time to complete the work-load. This is particularly true for QueenBee and to a lesser extent Ranger. Abe's performance is some where in between Ranger's and QueenBee's.

Next we perform experiments using more than one computational resource on the TG. To be precise, we employ more than one computational resource towards the solution of a single instance of the problem. Given that experiments are capable of utilizing three different TG resources, using two-resources at a time, gives us three different combinations. In figure 5, we present data when Ranger and QueenBee are used collectively. As a logical extension, we perform experiments wherein we utilize all three different TG resources. An important result that emerges is that, T_c decreases as the number of resources increases. This is demonstrated by the fact that, T_c (RQ) is lower than T_c (R) or T_c (Q). Although T_c (RQA) is less than T_c (RQ), the change is less significant than when going from one to two resources; however, performing many more experiments should lead to a clearer distinction in T_c. because of a single anamolous point in (R, Q, A). Finally, it is also important to point out that although T_c decreases, the overall number of CPU hours utilized does not increase.

In the next set of experiments, we employ BQP to autonomically decide which resources to utilize; we incorporate the decision making capability into the Lazarus framework. R-BQP represents the scenario (table 1), where BQP guides the selection of resource configuration on a predetermined machine (Ranger). When using more than one machine, e.g., RQA-BQP, both the selection of the resources and the selection of resource configuration are variables. For RQA-BQP, it is possible that even though three resources are available, all jobs be submitted to a single resource with much higher capacity or temporary lower load-factors (e.g, after a power-down/start-up). However, to correct for such corner cases and to take the more common case where all resources are essentially equal, we present data for RQA-BQP (table 2) when all three resources are utilized in the solution of the problem.

As shown in Figure. 5 (the second data-point from the left, and the right-most data-point), the utilization of BQP to determine which resources and which resource configuration are used, leads to a drastic reduction in T_c compared to the simple case where Lazarus framework was used without autonomic behaviour. This is a powerful demonstration of the fact that even elementary autonomic behaviour can lead to significantly improved performance.

# of Samples	Machine	Queue	Num. Cores	Duration (hrs)
3	Ranger	development	64	1:30
5	Ranger	development	64	2:00
2	Ranger	development	128	1:00
3	QueenBee	checkpt	16	2:00
4	QueenBee	checkpt	16	1:00
3	QueenBee	checkpt	32	1:00
3	Abe	dque	64	2:00
7	Abe	dque	64	1:00

Table 2: Table showing the selected configuration of the resources and the numer of times a particular configuration is chosen, when the decision is guided by BQP. Data in this table corresponds to RQA-BQP; the experiments are repeated ten times. As can be seen, the use of BQP results in a varying choice of resource configuration on different machines. In contrast, when BQP is not used, a fixed configuration is employed.

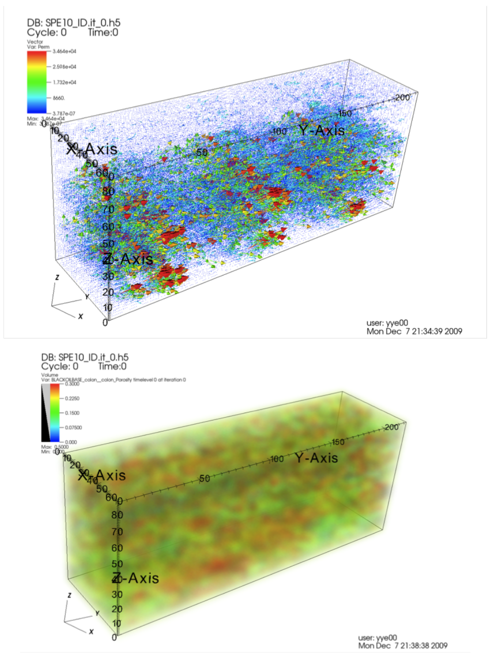


Figure 6: [Top] Reservoir as characterised by the permeability, and [Bottom] porosity.

In this work, in addition to a different scientific problem, we integrate (simulated) sensor data into the computational loop; the time-dependent sensor-data dynamically drives the application, and influences the controls and execution trajectory in phase-space.

These are amongst the first simulations that use general purpose frameworks (as distinguished from specialized frameworks, for example LEAD) on production infrastructure.

4.2 Reservoir Characterization

After several iterations of the workflow and the convergence of the EnKF to within a fixed value, a 3D map of the permeability (a measure of the flow of the rocks, figure 6 top) and porosity (a measure of the capacity of the rocks, figure 6 bottom) is output and defines the properties of the reservoir to a first approximation.

4.3 Analysis

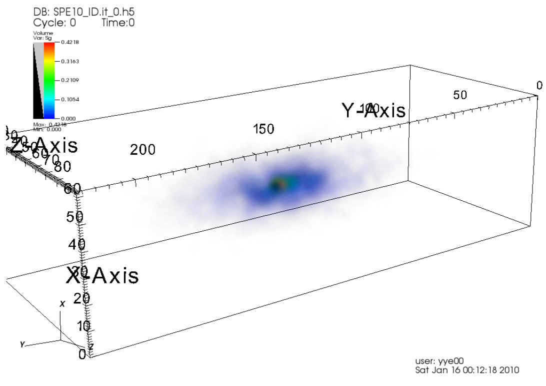


Figure 7: The saturation (relative proportion) of CO_2 for the reservoir characterised in Fig 6.

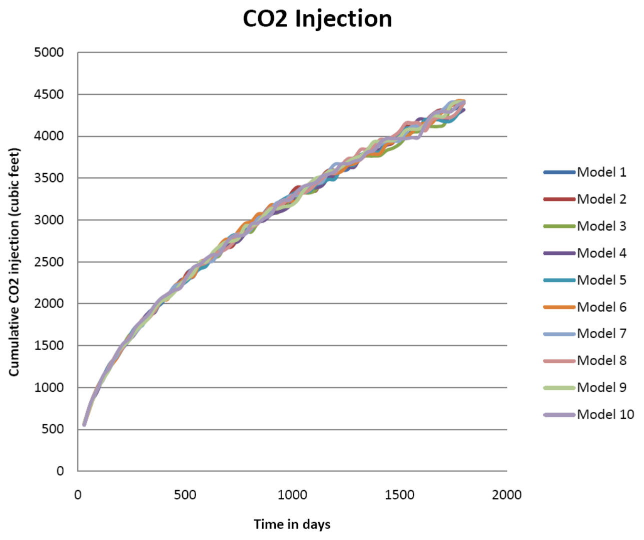


Figure 8: Simulation output of the injection of CO_2 as a function of time since start of injection. The output for 10 different models are plotted, showing the sensitivity of the injection amount to the underlying model.

The next step is to determine what is the extent of sequestration in a reservoir characterised by the porosity and permeability. The answer to this is provided by the relative proportion of CO_2 per grid cell. Figure 7 provides a measure of this relative proportion for the reservoir characterised in Figure 6. Fig 8 provides a quantitative estimate for how the injection varies with time for 10 rather similar models.

Conclusion: A triad of unique aspects are developed in this paper: (i) We incorporate the use of EnKF in reservoir characterization for carbon dioxide sequestration studies, combining two developing research areas and investigating the forefront (ii) We extend our EnKF based approach further with the incorporation of live sensor data that influences the entire study (iii) We establish the ability to *scale-out* to a large number of distinct heterogeneous resources.

The focus of this paper has been on developing the high-performance and distributed cyberinfrastructure required to simulate the CO_2 sequestration properties of a reservoir. The aim has not been to perform a detailed analysis or analyse the underlying scientific problem, but to derive the infrastructure requirements from the scientific requirements, to implement a solution and to validate our solution. This is consistent with an application-oriented cyberinfrastructure development philosophy.

5. REFERENCES

[1] SAGA Papers, 2003-2010. `http://saga.cct.lsu.edu/publications/papers/`.

[2] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.

[3] Cactus Framework. `http://www.cactuscode.org`.

[4] D. J. DePaolo et al. Basic research needs for geosciences:facilitating 21st century energy systems. *Office of Basic Energy Sciences, U.S. Department of Energy*, 2007.

[5] G. P. W. Don W. Green. *Enhanced Oil Recovery*, volume 6. 1998.

[6] R. Duff and Y. El Khamra. Real time simulation in grid environments: Communicating data from sensors to scientific simulations. In *Digital Energy Conference and Exhibition*. SPE, 2007.

[7] R. Duff and Y. El Khamra. A sensor and computation grid enabled engineering model for drilling vibration research. In *MG '08: Proceedings of the 15th ACM Mardi Gras conference*, pages 1–1, New York, NY, USA, 2008. ACM.

[8] G. Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] P. J. V. L. Gerrit Burgers and G. Evensen. Analysis scheme in the ensemble kalman filter. *Monthly Weather Review, American Meteorological Society*, 126, 1998.

[10] Y. Gu and D. S. Oliver. The ensemble kalman filter for continuous updating of reservoir simulation models. *Journal of Engineering Resources Technology*, 128(1):79–87, 2006.

[11] Y. Gu and D. S. Oliver. An iterative ensemble kalman filter for multiphase fluid flow data assimilation. *SPE Journal*, 12(4):438–446, 2007.

[12] P. H. Herschel L. Mitchell and G. Pellerin. Ensemble size, balance and model-error representation in an ensemble kalman filter. *Monthly Weather Review, American Meteorological Society*, 130, 2002.

[13] P. Houtekamer and H. L. Mitchell. Data assimilation using an ensemble kalman filter technique. *Monthly Weather Review, American Meteorological Society*, 126, 1998.

[14] S. Jha, H. Kaiser, Y. El Khamra, and O. Weidner. Design and implementation of network performance aware applications using saga and cactus. In *Accepted for 3rd IEEE Conference on eScience2007 and Grid Computing, Bangalore, India.*, 2007.

[15] R. E. Kalman. A new approach to linear filtering and prediction problems.

[16] Y. E. Khamra and S. Jha. Title: Developing Autonomic Distributed Scientific Applications: A Case Study From History Matching Using Ensemble Kalman-Filters. In *Sixth International Conference on Autonomic Computing, 2009. ICAC '09 (Barcelona)*. IEEE, 2009.

[17] X. Li, C. White, Z. Lei, and G. Allen. Reservoir model updating by ensemble kalman filter-practical approaches using grid computing technology. In *Petroleum Geostatistics 2007*, Cascais,Portugal, August 2007.

[18] H. Liu and M. Parashar, 2006. Accord: A Programming

Framework for Autonomic Applications.

[19] A. Luckow, S. Jha, J. Kim, A. Merzky, and B. Schnor. Adaptive distributed replica–exchange simulations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1897):2595–2606, 2009.

[20] L. Marini. *Geological Sequestration of Carbon Dioxide, Volume 11: Thermodynamics, Kinetics, and Reaction Path Modeling (Developments in Geochemistry)*, volume 11. 2007.

[21] B. S. Rajesh J. Pawar, Dongxiao Zhang and H. R. Westrich. Preliminary geologic modeling and flow simulation study of co2 sequestration in a depleted oil reservoir. *NETL 2001 Conference*, 2001.

[22] J. K. Shantenu Jha and Y. E. Khamra. Developing scientific applications with loosely-coupled sub-tasks, 2009.