World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:7, No:7, 2013

Parallelization of Ensemble Kalman Filter (EnKF) for Oil Reservoirs with Time-lapse Seismic Data

Md Khairullah, Hai-Xiang Lin, Remus G. Hanea, Arnold W. Heemink

Abstract—In this paper we describe the design and implementation of a parallel algorithm for data assimilation with ensemble Kalman filter (EnKF) for oil reservoir history matching problem. The use of large number of observations from time-lapse seismic leads to a large turnaround time for the analysis step, in addition to the time consuming simulations of the realizations. For efficient parallelization it is important to consider parallel computation at the analysis step. Our experiments show that parallelization of the analysis step in addition to the forecast step has good scalability, exploiting the same set of resources with some additional efforts.

Keywords—EnKF, Data assimilation, Parallel computing, Parallel efficiency.

I. Introduction

THE concept of a closed-loop model-based reservoir management, depicted in fig. 1, is a proper framework for reservoir monitoring and management [1]. It involves the use of (uncertain) reservoir and production models and combines them with available data from a real field. Data available from different sources can be used in the data assimilation loop to improve the characterization of the reservoir and the reliability of the flow predictions. Generally, data can be divided into two classes: *sparse data,* available only at the well locations and *dense data,* gathered everywhere in the reservoir. As the number of model parameters to be estimated is very large, the production history data or sparse data has a limited resolving power. On the other hand, due to the developments in geophysics, especially in the field of seismic, it is possible to track the fluids movements in the reservoir itself. This additional information such as *pressure (p)* and *saturation (s)*, in the form of *time-lapse seismic* or dense data, can be utilized together with production data, to narrow the solution space when minimizing the mismatch between gathered measurements and their forecasts from the numerical model [2]. In closed-loop reservoir management data assimilation is a vital part. The goal of this work is to enhance the performance of the data assimilation process by utilizing it's inherent parallelism.

The ensemble Kalman filter (EnKF) has gained popularity in data assimilation for non linear systems in the recent years. However, EnKF takes large turn around time in terms of computing, particularly to run the simulations of hundreds of realizations. The situation worsens when we have large number of observations, which makes the analysis step time consuming. But fortunately enough, both the forecast and the analysis parts of EnKF have inherent parallelism. The forecast step is embarrassingly parallel as each ensemble member can run independently in separate processing element of computers. The analysis or update step is critical for only systems with large number of observations and parallel implementation of this step is not simple because of the intermediate inter process communication requirement.

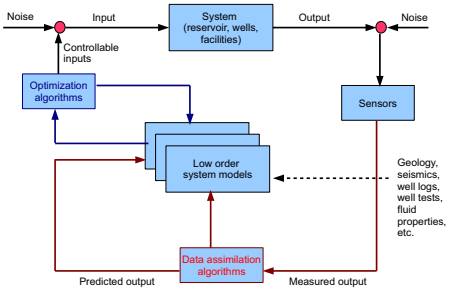


Fig. 1: Reservoir management depicted as a closed loop model-based controlled process [1]

Data assimilation experiments are performed using a parallel ensemble Kalman filter (EnKF) in [3], [4], [5], [6], [7], [8], and [9] for various applications of atmospheric data analysis, oceanography and hydrology. The common feature of these parallel implementations is that they only consider the total parallelization of the forecast step. Parallelization of the analysis step is either completely ignored or done partially or incorporates some sort of domain decomposition.

Parallelization of other variants of EnKF or closely related approaches are also reported. A parallel implementation of EnKF based on the Sherman-Morrison-Woodbury formulations is presented in [10]. A parallel ensemble adjustment Kalman filter (EAKF) has been designed and implemented using a local least squares framework for El Nino–Southern Oscillation (ENSO)) forecast system in [11]. A variant of a least squares ensemble (Kalman) filter is implemented on parallel architectures in [12]. [13] presents a parallel implementation of Singular Evolutive Interpolated Kalman (SEIK) filter. The scalability of different ensemble-based Kalman filters are discussed in [14]. Again, the forecast step in the above works is implemented by running each realization or

Md Khairullah is with the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet, 3114, Bangladesh(e-mail: khairullah-cse@sust.edu).
Hai-Xiang Lin is with Delft Institute of Applied Mathematics (DIAM), TU Delft, Netherlands.(e-mail: H.X.Lin@tudelft.nl).
Remus G. Hanea is with Delft Institute of Applied Mathematics (DIAM), TU Delft, Netherlands.(e-mail: R.G.Hanea@tudelft.nl).
Arnold W. Heemink is with Delft Institute of Applied Mathematics (DIAM), TU Delft, Netherlands.(e-mail: A.W.Heemink@tudelft.nl).

ensemble member independently on each processor. However, the analysis step is enhanced by reducing inter process communication applying some form of *domain decomposition* or *regionalization* to avoid allocating and loading large state vectors from each processor in most of the implementations. Issues such as smoothness of the analysis fields across the sub-domain boundaries is a concern in domain localization for parallelization of the ensemble algorithms and are discussed in [15].

In [16] and [17] parallel implementations of EnKF for oil reservoir are presented. The first level of parallelization is usually during the forecast step. A second level of parallelization is implemented by a parallel reservoir simulator for each realization. The third level is achieved by partially parallelizing the analysis step. A parallel framework for history matching and uncertainty characterization based on EnKF and ensemble smoother (ES) for the application of reservoir simulation is presented in [18]. [19] presents development of a high–performance grid computing environment that performs the forecast step of the EnKF algorithm in parallel by simultaneous run of each simulation model.

No previous parallel EnKF is reported to attempt parallelizing the analysis step as a whole. For the first time we parallelize the total analysis step with the whole domain in this work. Moreover, no previous parallel EnKF for oil reservoir is found to deal with dense data. Our parallel EnKF integrates dense data from time lapse seismic for oil reservoirs. This paper presents a parallel EnKF for oil reservoirs data assimilation and the experimental results. The results show that by parallelization of the whole EnKF process, significant speed up can be achieved utilizing the same set of parallel hardware, which are used for the forecast step only or the forecast step and partially for the analysis step in some previous works. In the next section we briefly discuss data assimilation concepts along with Kalman filter and ensemble Kalman filter. Then we describe some implementation details. After presentation and analysis of the results we conclude with discussion of future research directions.

II. Data Assimilation or History Matching

For any dynamic system, there are two sources of information: the measurements of the system or *observations* and the understanding of the temporal and spatial evolution of the system or *models*. The state estimation problem is defined as finding the estimate of the model state that best fits the model equations, the initial and boundary conditions, and the observed data in some weighted measure [20].

A. Ensemble Kalman Filter

The classical Kalman Filter (KF) updates the state estimate whenever measurements are available, using the forecast estimates, error covariance of the predicted model state, and the measurement error covariance [21].

However, results of rigorous solutions to the non-linear problem by Kalman filter are either too narrow in applicability or are computationally expensive [22]. The ensemble Kalman filter is a Monte Carlo [23] approach for approximating Kalman filter to overcome these limitations. In the analysis step in EnKF, updates are performed on each of the ensemble members and is given by

$$\psi_j^a = \psi_j^f + (C_{\psi\psi}^e)^f M^T (M(C_{\psi\psi}^e)^f M^T + C_{\epsilon\epsilon}^e)^{-1}(d_j - M\psi_j^f), \quad (1)$$

and is expressed in terms of the ensemble matrices as

$$A^a = A + C_{\psi\psi}^e M^T (MC_{\psi\psi}^e M^T + C_{\epsilon\epsilon}^e)^{-1}(D - MA), \quad (2)$$

where the measurement matrix is $M \in \mathbb{N}^{m\times n}$. These can be further reduced [24] to the EnKF update equation

$$A^a = AX, \quad (3)$$

where the matrix A is the ensemble of the forecasted state vectors, A^a is the ensemble of the updated state vectors and X, let us call it the updating coefficient matrix, is computed in the analysis step based on the forecast and the measurements, to minimize the variances of the state vectors among the ensemble members. So the EnKF analysis step consists of construction of the update coefficient matrix X, and the matrix multiplication of A and X in (3).

B. Parallelization of EnKF

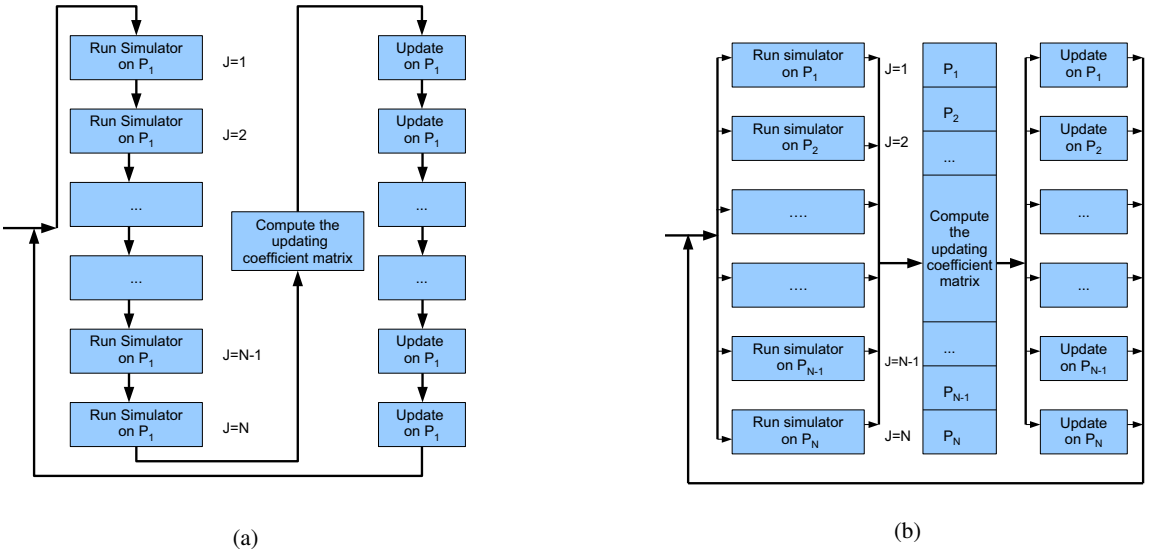
In [16] and [17], the first two levels of parallelization work for the construction of the matrix A and the third level is used only for the matrix-matrix multiplication in (3).

However, the construction of the matrix X in (3) would be very compute intensive for large number of observations. The construction of X involves a series of singular value decomposition (SVD) computation and matrix-matrix multiplications and these two operations can be carried out efficiently in parallel computers due to availability of efficient parallel algorithms and corresponding libraries. Hence, also the construction of X can be parallelized and significant reduction in assimilation time can be obtained. Provided parallel computing resources for running the simulations in parallel, we can exploit them for the large computations for the analysis steps. As we said, the matrix X is termed as the updating coefficient matrix in the flowcharts in fig. 2. By the statement 'update' in the flowcharts we mean the matrix-matrix multiplication in (3).

Fig. 2 illustrates the differences in work flows of a serial and the proposed parallel implementation of the method. In these examples the EnKF method works with N realizations. In the serial implementation, all realizations and the analysis step are executed on a single processor P_1, whereas in the parallel counterpart each realization runs on a separate processor independently, the analysis step run on N processors in a mixture of coordinated and independent fashion.

III. Implementation

Inclusion of time-lapse seismic data makes the EnKF process more time consuming as the analysis step becomes larger than the forecast step in terms of execution time. In addition to this, the analysis part is more complex than the forecast step as it requires intermediate inter process communications. By intelligent use and runtime configurations of the ScaLAPACK



(a) (b)

Fig. 2: Implementation of EnKF for data assimilation: (a) serial version, (b) proposed parallel version

routines we minimize the requirement of data communications effectively and achieve very good speedup values. Even with large number of processors, our implementation shows better scaling.

A. Reservoir model

We work with a rectangular reservoir field with length 1980 m, width 1020 m and height 5 m. We assume all the fluid and rock properties of the reservoir to be constant along the height and hence we consider it as a two dimensional reservoir. For computational purpose the field is logically divided into 5049 blocks in a 51×99 grid of square blocks of size 20 m in each side. We have two injection wells located at the coordinates (2, 2) and (50, 17) and two production wells located at the coordinates (17, 62) and (41, 98)

The typical state vector for oil reservoir consists of the variables: (i) *pressure* p, (ii) *saturation* s, (iii) *bottom hole pressures in the wells BHP*, (iv) *oil flow rate in the wells* q_o, and (v) *water flow rate in the wells* q_w. We consider observations of saturations (in terms of time-lapse seismic) at every block and bottom hole pressures, and oil and water flow rates in the wells (in terms of measurements). We consider the following standard deviations of the measurement noise (Gaussian) for observations: 15% for saturation, 5% for BHP and oil flow rate, and 10% for water flow rate. We want to estimate the *log permeability log k* also. Then our state vector has the following form.

A=$\left[\ p,\ s,\ BHP,\ q_o,\ q_w,\ log\ k\ \right]^T$

We assume a constant porosity field in fig. 3 for all ensemble members and subsequently porosity is not part of the analysis process.

Our measurement network excludes pressure values and permeability values. Then for computations in the analysis steps, we use the following portion (with length 5061) of the state vector:

A=$\left[\ s,\ BHP,\ q_o,\ q_w\ \right]^T$

And as estimating the log permeability values ($log\ k$) is our primary goal of the assimilation and updating other state values are of minor importance (and we neglect them) and hence in the update statement in (3) we use only the following portion (of length 5049) of the state vector.

A=$\left[\ log\ k\ \right]$

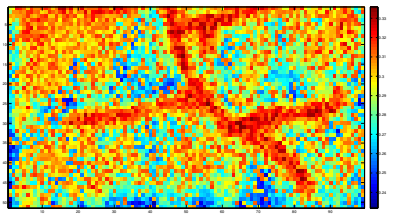


Fig. 3: Assuming constant porosity field for all ensemble members

B. Software structure

For the forecast or simulation step we used *simsim* (simple simulator), written in MATLAB, developed by Prof. Jan Dirk Jansen at TU Delft. For parallel communication in data assimilation algorithms we used MPI in C language. To interface with simsim we developed a driver named *simsimdriver* which contains all data assimilation computations as well as function calls for running the simulator for the forecast step. Two wrapper functions were developed in MATLAB: (i) *initSimsim,* which initializes and prepares all the required resources such as model, control and parameter informations on each processor and provides assumed permeabilities of the ensemble, and (ii) *runSimsim,* which works with the updated

assumption of permeabilities for a defined time period and returns the seismic data for saturation and measurement data of bottom hole pressure, and oil and water flow rates measured in the wells. These wrapper functions and the original simsim functions are compiled to a library, usable in C programming, by the MATLAB compiler *mcc* tool with appropriate settings. The driver program in C and the compiled library needs to be compiled to an executable with the MATLAB compiler *mbuild* utility.

C. Optimization

We adopt the two most common ways to optimize parallel algorithms, load balancing and communication minimization, and discuss below.

We statically distribute equal number of realizations to each processor to run the simulation. There is no processor-to-processor communication during the forecast step, but it requires a very complex dynamic load balancing in this step. In order to obtain good load balancing in the analysis step we choose small block sizes of the operand matrices for ScaLAPACK operations. We choose block size 64 in case of the long dimensions and $\frac{N}{nproc}$ in case of short dimensions, where $nproc$ is the number of processes. In fact the smallest possible block size 1 ensures 100% load balancing, but then the number of communications for ScaLAPACK operations is large, leading to bad performance. ScaLAPACK suggests a 64×64 two dimensional blocks for a balanced load as well as tolerable communication overhead for optimum performance.

Other than the internal communications of the ScaLAPACK operators, we put significant efforts to minimize data communication by two ways: (i) construction and computations of the local matrices locally and (ii) using non blocking communication where possible.

IV. Results and Discussions

We tested our implementation on two different clusters, first one is in a local area network (LAN) with heterogeneous hardware and the other one is the SARA Lisa high performance compute cluster. In both clusters, we used a single core of each processor to have homogeneous data communication among all the processing elements.

A. Verification

The main goal of data assimilation is to minimize the difference between the predictions and the true values for the reference variables or parameters. With time (or assimilation steps) the difference should reduce in general. Root mean square (*rms*) difference between the assimilated parameters with the true values is a good tool to perceive the result.

In our experiments, the models simulate the reservoir operations for ten years, whereas data assimilations take place in the first three years at six months interval, resulting in six assimilation steps. Fig. 4 shows the rms difference of the assimilated log permeability and true log permeability when executed on four processors with 16, 48, 96, 144 and 288 realizations. With large number of realizations, the EnKF performs better. Additionally we observe that with increased number of realizations the performance of the EnKF does not improve linearly.

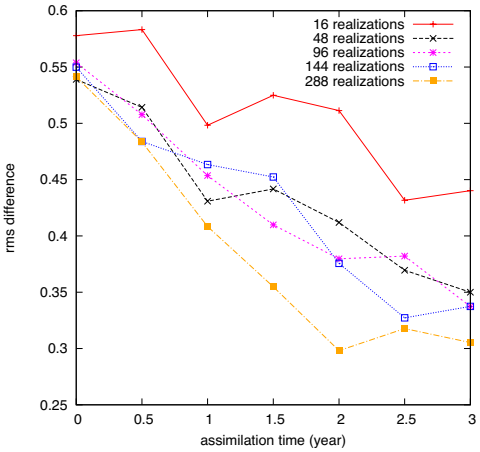


Fig. 4: rms differences of log permeability over time

Fig. 5 presents the log permeability graphs for visual comparison.

B. Performance of the parallel algorithm

Initial tests and measurements were carried out in a LAN cluster with heterogeneous processing speed (2.20-3.33 GHz) and memory size (2-6MB cache memory and 4-8 GB main memory) configurations. For the single processor run we used the 2.66 GHz Intel(R) Core(TM)2 Quad CPU Q8400 machine with 6MB cache and 4 GB main memory (i.e. roughly the average configuration). The LAN has a data rate of 100 Mbps. Then measurements were taken in the SARA Lisa cluster having 8 core processors with 2.26 GHz speed, 8 MB cache and 24 GB main memory, and 4x DDR Infiniband network with bandwidth 1600 MB/sec and Latency < 6 μsec.

In following discussion the forecast time, analysis time and the total time are related as

$$total\ time = initialization\ time + forecast\ time + analysis\ time. \quad (4)$$

Table I and II show the time measurements, corresponding speedup and parallel efficiency information on with 48 and 96 realization respectively. Fig. 6 and fig. 7 depict the speedup on both the clusters.

1) Difference in speedup: forecast vs analysis step: The forecast step need no intermediate inter process communication and hence the interconnection network has no role at all in the speedup of this step and the speedup follows the ideal case. On the other hand in the computation of SVD and matrix-matrix multiplication, naturally we need a huge number of intermediate inter process communication and this implies that the speedup in the analysis step will be less than that of the forecast step.

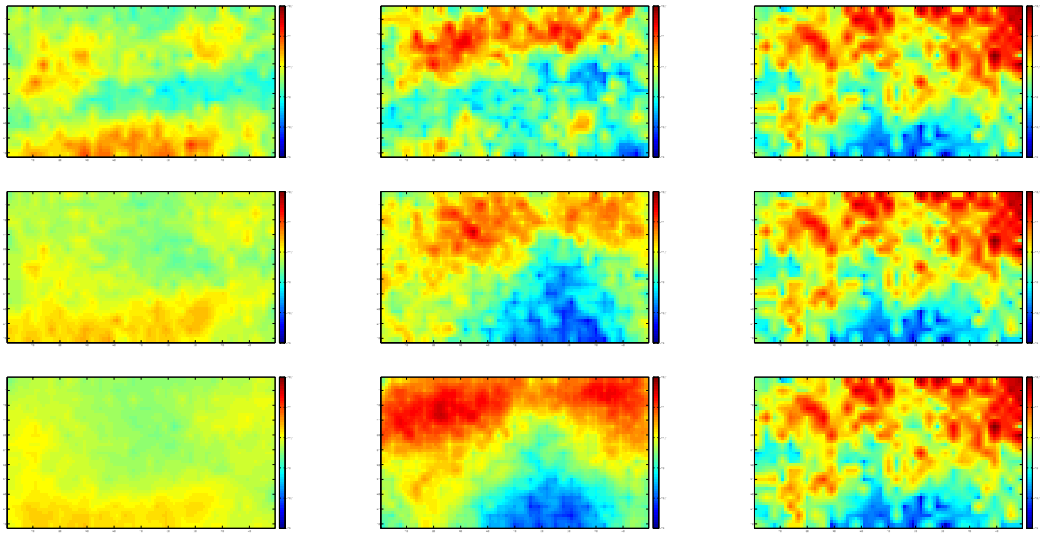
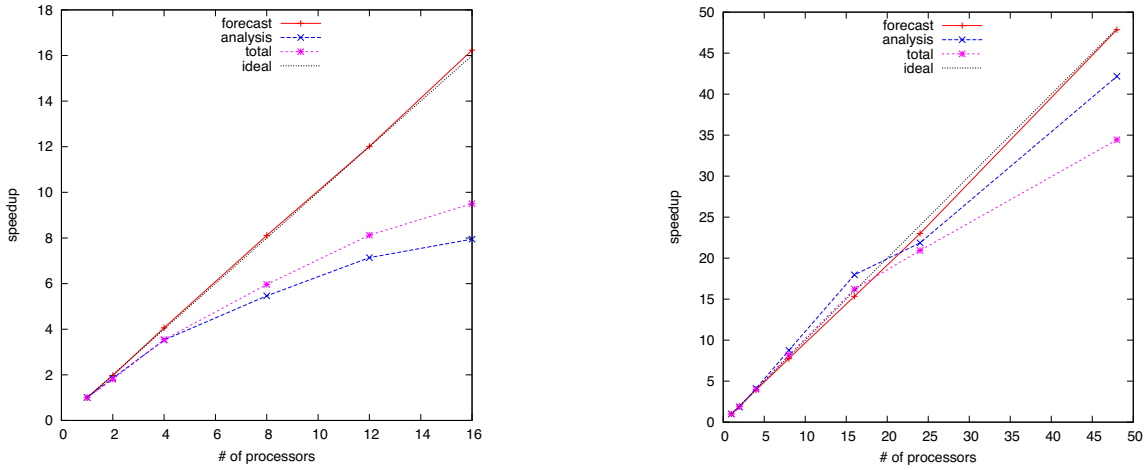


Fig. 5: log permeability fields with 16 (top row), 48 (middle row) and 288 (bottom row) realizations: initial (left column), after 6 assimilation steps in 3 years (middle column), the true log permeability (right column)

TABLE I: Performance of the parallel implementation with 48 realizations

# processors	1		4		8		16		48
cluster	LAN	Lisa	LAN	Lisa	LAN	Lisa	LAN	Lisa	Lisa
forecast time (h)	2.604	2.99	0.75	0.75	0.37	0.39	0.19	0.2	0.06
analysis time (h)	4.62	3.84	1.31	0.94	0.85	0.44	0.58	0.21	0.09
total time (h)	7.66	6.85	2.16	1.71	1.29	0.84	0.81	0.42	0.20
S_P (forecast)	1	1	4.06	3.97	8.11	7.73	16.23	15.34	47.86
S_P (analysis)	1	1	3.53	4.1	5.46	8.76	7.95	17.97	42.17
S_P (total)	1	1	3.54	4	5.96	8.19	9.51	16.22	34.43
E_P (forecast) %	100	100	101.48	99.14	101.36	96.66	101.46	95.85	99.70
E_P (analysis) %	100	100	88.13	102.47	68.25	109.47	49.69	112.31	87.85
E_P (total) %	100	100	88.59	99.91	74.51	102.37	59.41	101.36	71.72



(a) (b)

Fig. 6: Speedup of the parallel implementation for 48 realization on (a) LAN cluster and (b) SARA Lisa cluster for different parts

TABLE II: Performance of the parallel implementation with 96 realizations

# processors	1		4		8		16		48
cluster	LAN	Lisa	LAN	Lisa	LAN	Lisa	LAN	Lisa	Lisa
forecast time (h)	6.22	6.1	1.52	1.54	0.98	0.8	0.49	0.37	0.12
analysis time (h)	4.13	3.72	1.41	1	0.78	0.43	0.55	0.21	0.09
total time (h)	10.37	9.84	3.15	2.608	1.77	1.24	1.06	0.62	0.24
S_P (forecast)	1	1	4.09	3.97	6.37	7.63	12.68	16.43	49.93
S_P (analysis)	1	1	2.92	3.74	5.30	8.76	7.46	17.44	41.09
S_P (total)	1	1	3.29	3.81	5.84	7.91	9.76	15.8	41.67
E_P (forecast) %	100	100	102.31	99.26	79.68	95.33	79.25	102.7	104.02
E_P (analysis) %	100	100	72.602	93.5	66.23	109.5	46.65	109	85.61
E_P (total) %	100	100	82.37	95.19	72.605	98.81	60.98	98.75	86.82

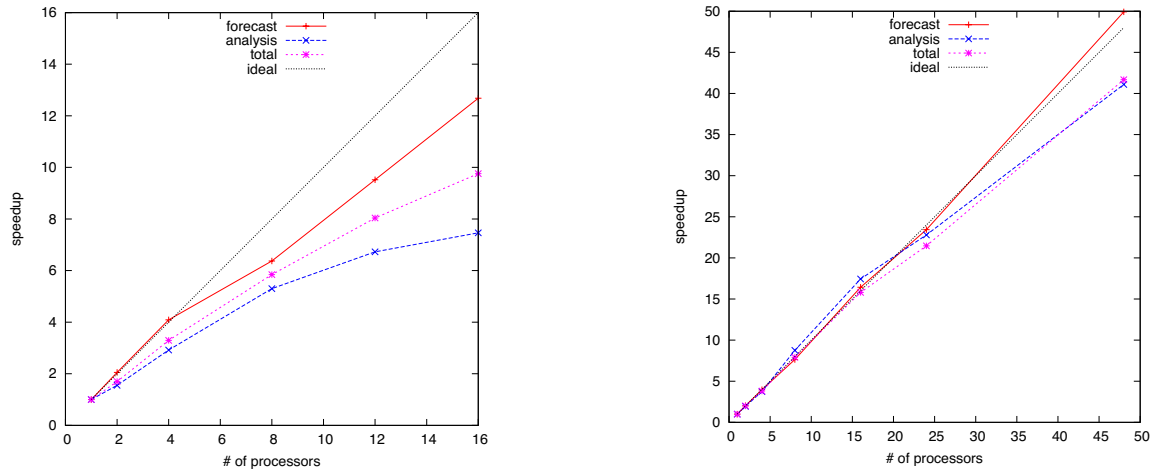


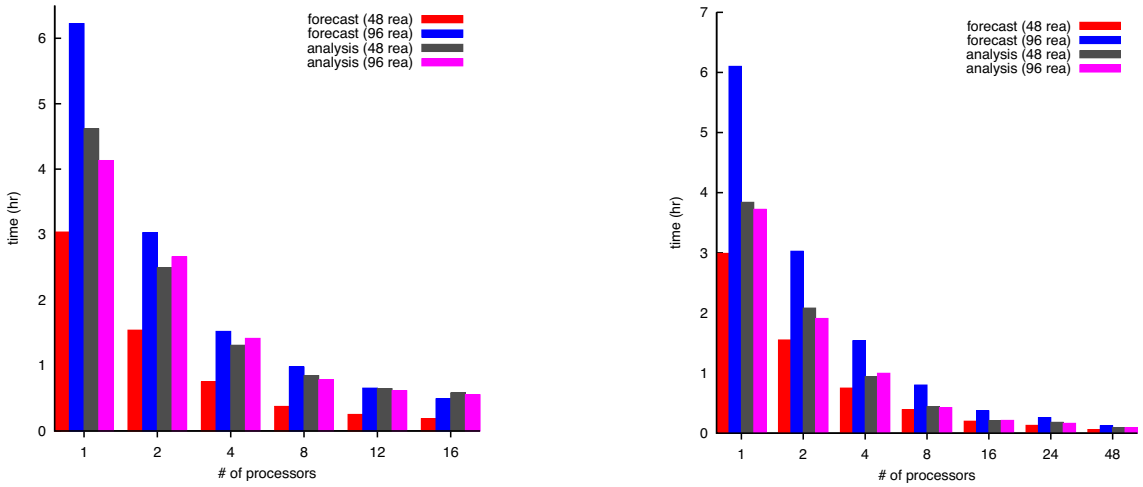
Fig. 7: Speedup of the parallel implementation for 96 realization on (a) LAN cluster and (b) SARA Lisa cluster for different parts

2) Super-linear speedup: In both the clusters super-linear speedup is observed for the forecast step. In the SARA Lisa cluster we observe this also for the analysis step. This occurs when number of processors becomes large enough such that their combined cache memory size is enough to hold the required data which is impossible for a single or small number of processors. In the Lisa cluster every processor has a cache memory of 8 MB, whereas in the LAN cluster the maximum available cache memory size is 6 MB and the minimum is 2 MB. The shortfall in the cache memory size and the mentioned slow interconnection network do not allow speedup to be super-linear for the analysis step in the LAN cluster.

3) Parallel scalability: For both 48 and 96 realizations on both the clusters, the efficiencies of the forecast steps almost always follow the linear speed up. This implies that the forecast step is strongly scalable regardless of the interconnection network of the cluster. In the forecast step with 96 realization the problem size is double of that with 48 realizations. This means the forecast step is also weakly scalable. However, for both 48 and 96 realizations on both the cluster the efficiencies of the analysis steps gradually decrease with increased number of processors. This implies that the analysis step is not strongly scalable. For proper scaling of SVD computation and matrix-matrix multiplication, ScaLAPACK suggests at least a 1000×1000 matrix per processor after data distribution. By this formulation, 25 is the roughly calculated upper limit of number of processors for the model we are working with to get good efficiency for the analysis step. We observe both for 48 and 96 realizations, on the SARA Lisa cluster the efficiency is well above 90% for up to 24 processors.

4) Difference in performance: 48 realizations vs 96 realizations: With more number of realizations the problem size for the forecast step increases proportionally. These are observed in fig. 8. In both cases, the forecast time for 96 realizations is almost double of that for 48 realizations. On the other hand, the problem size for the analysis part remains almost the same for varying number of realizations, as the dimension of matrix in the SVD computation and the large matrix multiplication immediately after this remain the same ($m\times m$ or 5061×5061 for our working model).

5) Difference in speedup and performance: LAN cluster vs SARA Lisa cluster: The forecast step needs no intermediate inter process communication. So the speedup of this step should not be affected by the interconnection network of the used cluster. We see in the figures, for both clusters the speedup of the forecast step follows almost the ideal case. However, as SARA Lisa uses faster interconnection network, the effect of the intermediate inter process communications at the analysis step should be less severe to it, compared to the LAN cluster and the figures show that for the SARA



(a) (b)

Fig. 8: Execution time of the forecast and analysis step for 48 and 96 realizations on (a) LAN cluster and (b) the SARA Lisa cluster

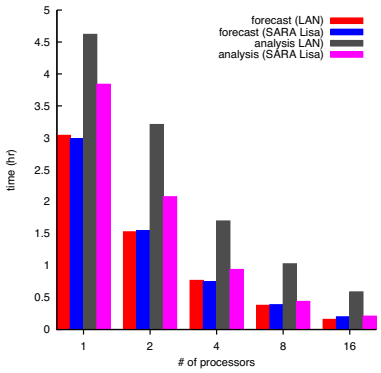


Fig. 9: Comparison of execution time of the forecast and analysis step for 48 realizations on different clusters

Lisa cluster, the analysis step speedup is slightly less than the forecast step speedup. We see in the fig. 9 the forecast time is almost equal in both clusters, but as the SARA Lisa cluster uses faster interconnection network, it requires very less time for analysis step compared to the LAN cluster.

6) Effect of ScaLAPACK process grid: Process grid arrangement of the available processors is an important performance issue for ScaLAPACK operations. ScaLAPACK suggests a square grid or in case it is impossible to construct a square grid then very close to a square grid arrangement. The performance issue of the process grid arrangement is more vital for slow interconnection networks due to communication requirements. Table III lists the various analysis times for possible 3 process grid arrangements of 16 processors on both clusters. Fig. 10 depicts the data in the table. We observe that the 1×16 grid, which is the worst arrangement compared to a square grid arrangement, shows the worst performance whereas the 4×4 grid shows the best performance.

TABLE III: Execution time (in minute) of the analysis step for different process grid orientation for 48 realizations

process grid	LAN cluster	SARA Lisa cluster
1x16	77.43	15.97
2x8	44.48	13.78
4x4	35.66	12.83

7) Effect of ScaLAPACK block size: ScaLAPACK logically divides the data in the original matrix in two dimensional blocks and distributes the matrix in units of these blocks among the available processors in the process grid. A block with length and width 1, that is a block with just 1 element ensures the highest load balancing of the ScaLAPACK operations but increases the inter process communication for ScaLAPACK operations. So there is a trade-off between load balancing and the amount of data communication for the optimal block length and width. Though ScaLAPACK suggests 64 to be the optimal choice, with available larger cache memory size, the optimal block length may vary for different applications. Table IV lists the analysis time for different block lengths on both clusters. We applied this only to the long dimensions (with length 5061 or 5049 for the used model) of the matrices described earlier. Fig. 11 reports the execution time of the analysis step as a function of block sizes. Varying lengths of the block has a less affect on the performance compared to the variation of process grid arrangements. Also the LAN cluster with slow interconnection network is more sensitive to the variation in the block lengths.

V. Conclusion

The parallel EnKF for oil reservoirs developed in this work possesses the following key properties: (i) parallelization of the whole EnKF process; (ii) use of sparse measurement data as well as dense time-lapse seismic data; (iii) no localisation;

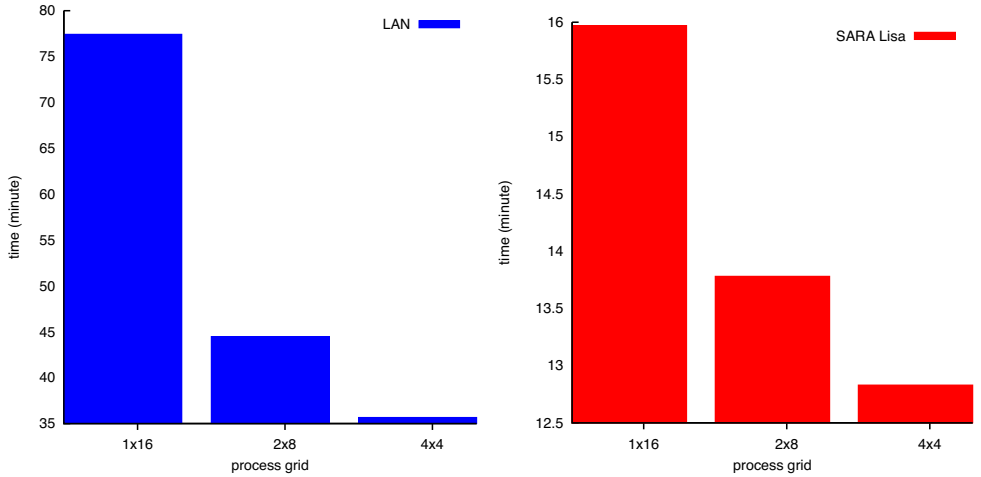


Fig. 10: Execution time of the analysis step for different process grid orientation for 48 realizations

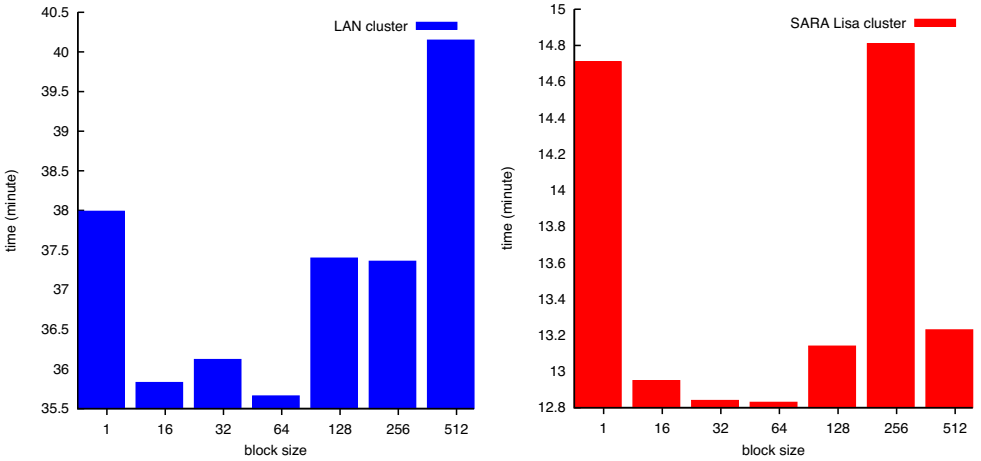


Fig. 11: Execution time of the analysis step for different block sizes for 48 realizations

TABLE IV: Execution time (in minute) of the analysis step for different block sizes for 48 realizations

Block Length	SARA Lisa Cluster	LAN Cluster
1	14.71	37.99
16	12.95	35.83
32	12.84	36.12
64	12.83	35.66
128	13.14	37.40
256	14.81	37.36
512	13.23	40.15

TABLE V: Performance comparison between our work and parallel EnKF

Parallel EnKF	realizations	processors	efficiency
R. Tavakoli et al.	100	50	50.8% and 47.2%
Our EnKF	96	48	86.8%
B. Liang et al.	200	16	76.8%
Our EnKF	96	16	98.8%

(iv) minimization of data communications by intelligent use and runtime configurations of ScaLAPACK routines.

It outperforms the previous parallel implementations of EnKF for oil reservoirs in terms of parallel efficiency. Table V shows the comparison of our work with previous works Important conclusions regarding performance are: (i) parallelization of EnKF is limited by the analysis step; (ii) the forecast step shows linear speedup (occasionally super-linear speedup) and scalability; (iii) the analysis step shows sub-linear speedup and super-linear speedup in fast interconnection clusters for small number of processors due to cache; (iv) the number of realizations impacts the forecast step time, whereas the analysis step time remains almost the same; (v) the forecast time is almost independent of the hardware and interconnection network; (vi) the analysis time significantly depends on the interconnection network, and is affected by ScaLAPACK process grid organization, where ScaLAPACK block length has a minor effect.

For further performance gain parallel EnKF can be implemented on GPU clusters. We need to develop CUDA

kernels for expensive functions of *simsim*. Or other GPU implementations of oil reservoirs can be used for simulation. For the analysis step we need GPU versions of singular value decomposition and matrix-matrix multiplication of ScaLAPACK equivalent routines.

ACKNOWLEDGMENT

The authors would like to thank SARA, the Netherlands Supercomputing Centre, Amsterdam for providing the computing facility *the Lisa cluster*.

REFERENCES

[1] J. D. Jansen and S. D. Douma and D. R. Brouwer and P. M. J. Van den Hof and O. H. Bosgra and A. W. Heemink. Closed loop reservoir management. In SPE Reservoir Simulation Symposium, The Woodlands, Texas, U.S.A., February 2009. Society of Petroleum Engineers.

[2] S. Gillijns and O. BarreroMendoza and J. Chandrasekar and B. L. R. DeMoor and D. S. Bernstein and A. Ridley. What is the ensemble Kalman filter and how well does it work? In Proceedings of the 2006 American Control Conference, pp. 4448-4453, 2006.

[3] Herschel L. Mitchell and P. L. Houtekamer, An Adaptive Ensemble Kalman Filter, MONTHLY WEATHER REVIEW, 128(2): 416–433, February 2000.

[4] P. L. Houtekamer and Herschel L. Mitchell, A Sequential Ensemble Kalman Filter for Atmospheric Data Assimilation, MONTHLY WEATHER REVIEW, 129(1):123–137, JANUARY 2001.

[5] Christian L. Keppenne. Data assimilation into a primitive-equation model with a parallel ensemble Kalman filter. MONTHLY WEATHER REVIEW, 128(6):1971-1981, June 2000.

[6] Christian L. Keppenne and Michele M. Rienecker. Initial testing of a massively parallel ensemble Kalman filter with the poseidon isopycnal ocean general circulation model. MONTHLY WEATHER REVIEW, 130(12):2951-2965, December 2002.

[7] Christian L. Keppenne and Michele M. Rieneckerb, Assimilation of temperature into an isopycnal ocean general circulation model using a parallel ensemble Kalman filter, Journal of Marine Systems, 40-41 (2003), pp. 363380.

[8] Teng Xua and J. Jaime Gomez-Hernandeza and Liangping Lia and Haiyan Zhoua, Parallelized Ensemble Kalman Filter for Hydraulic Conductivity Characterization, Computers & Geosciences, 52: 42-49, March 2013.

[9] Edward Ott and Brian R. Hunt and Istvan Szunyogh and Aleksey V. Zimin and Eric J. Kostelich and Matteo Corazza and Eugenia Kalnay and D. J. Patil and James A. Yorkey, A local ensemble Kalman filter for atmospheric data assimilation, Tellus (2004), 56A, pp. 415-428.

[10] Jan Mandel, Efficient Implementation of the Ensemble Kalman Filter , CCM Report 231, May 2006, http://math.ucdenver.edu/ccm/reports/rep231.pdf

[11] S. Zhang and M. J. Harrison and A. T. Wittenberg and A. Rosati and J. L. Anderson and V. Balaji, Initialization of an ENSO (El NinoSouthern Oscillation (ENSO)) Forecast System Using a Parallelized Ensemble Filter, MONTHLY WEATHER REVIEW, 133: 3176–3201.

[12] Jeffrey L. Anderson and Nancy Collins, Scalable Implementations of Ensemble Filter Algorithms for Data Assimilation, JOURNAL OF ATMOSPHERIC AND OCEANIC TECHNOLOGY, 24(8): 1452–1463, AUGUST 2007.

[13] Lars Nerger and Wolfgang Hiller, Software for ensemble-based data assimilation systems - Implementation strategies and scalability, Computers & Geosciences, Available online 7 April 2012.

[14] Lars Nerger and Wolfgang Hiller and Jens Schroeter, PDAF - the Parallel Data Assimilation Framework: Experiences with Kalman filtering. In: Zwieflhofer, W., Mozdzynski, G. (Eds.), Use of High Performance Computing in Meteorology - Proceedings of the 11. ECMWF Workshop. World Scientific, pp. 63-83.

[15] Janji Tijana and Lars Nerger and Alberta Albertella and Jens Schroeter and Sergey Skachko, On Domain Localization in Ensemble-Based Kalman Filter Algorithms, Monthly Weather Review, 139: 2046-2060, 2011.

[16] B. Liang and K. Sepehrnoori and M. Delshad, An Automatic History Matching Module with Distributed and Parallel Computing, Petroleum Science and Technology, 27(10): 1092–1108, January 2009.

[17] Reza Tavakoli and Gergina Pencheva and Mary F. Wheeler. Multi-level parallelization of ensemble Kalman filter for reservoir history matching. In 2011 SPE Reservoir Simulation Symposium.

[18] Reza Tavakoli and Gergina Pencheva and Mary F. Wheeler and Benjamin Ganis, A parallel ensemble-based framework for reservoir history matching and uncertainty characterization, Computational Geosciences, 17(1): 83–97, February 2013.

[19] L. Xin, Continuous Reservoir Model Updating by Ensemble Kalman Filter on Grid Computing Architectures, Ph.D. thesis, Louisiana State University, Baton Rouge, Louisiana, 2008.

[20] G. Evensen.The Ensemble Kalman Filter for combined state and parameter estimation, Monte Carlo techniques for data assimilation in large systems. IEEE CONTROL SYSTEMS MAGAZINE, 2009.

[21] R. E. Kalman. A new approach to linear filter and prediction problems. J. Basic Eng., 82:35-45, 1960.

[22] S. Gillijns and O. BarreroMendoza and J. Chandrasekar and B. L. R. DeMoor and D. S. Bernstein and A. Ridley. What is the ensemble Kalman filter and how well does it work? In Proceedings of the 2006 American Control Conference, pages 4448-4453, 2006.

[23] Eric W. Weisstein, Monte carlo method, http://mathworld.wolfram.com/MonteCarloMethod.html, Last visited on 26-02-2013.

[24] G. Evensen. Data Assimilation: The Ensemble Kalman Filter. Springer: New York, 2007.