On the Problem-Decomposition of Scalable 4D-Var Data Assimilation Models

R. Arcucci
University of Naples Federico II, Naples (IT)
Imperial College London, London (UK)
r.arcucci@imperial.ac.uk

L. D'Amore
University of Naples Federico II, Naples (IT),
Euro Mediterranean Center on Climate Changes
Lecce, (IT)
luisa.damore@unina.it

L. Carracciuolo
National Research Council
Naples, (IT)
luisa.carracciuolo@cnr.it

***Abstract*—We present an innovative approach for solving Four Dimensional Variational Data Assimilation (4D-VAR DA) problems. The approach we consider starts from a decomposition of the physical domain; it uses a partitioning of the solution and a modified regularization functional describing the 4D-VAR DA problem on the decomposition. We provide a mathematical formulation of the model and we perform a feasibility analysis in terms of computational cost and of algorithmic scalability. We use the scale-up factor which measure the performance gain in terms of time complexity reduction. We verify the reliability of the approach on a consistent test case (the Shallow Water Equations).**

***Keywords*—Data Assimilation, Problem Decomposition, Scalable Algorithm, Inverse Problem, Ocean Models.**

I. Introduction and Motivation

The goal of Data Assimilation (DA) is to obtain an optimal estimate of the state of the system (atmosphere, ocean,...) from a large set of observations and prior information, at a given time. One of its primary use is to provide a direct confrontation between numerical model predictions with actual observations, i.e., incorporating the actual observations into numerical weather prediction and/or climate modeling. Data assimilation also is used in re-analyses, which are important tools to evaluate climate simulations [5].
Current 4D-Var DA techniques are not geared towards exascale computing. Indeed, 4D-Var is a highly sequential algorithm, since iterations of the minimisation algorithm are sequential, the Adjoint integrations run one after the other, and the Model time steps follow each of the other. Up to now, main efforts towards the development of scalable 4D Var DA systems were achieved in numerical weather prediction applications, namely by NCAR (National Center for Atmospheric Research), in Colorado (USA), by distributing the grid points over processors [16], and by the ECMWF (European Centre for Medium-Range Weather Forecasts), in Reading (UK), where the saddle point formulation of weak-constraint 4D-Var DA algorithm allows parallelism in the time dimension [7]. Finally, it is worth to cite the parallel 4D-Var DA module of the Regional Ocean Modeling System (ROMS), developed by University of California, USA, where parallelism is introduced in the code by distributing the grid points over processors [12], [13].
Our innovative approach is based on a mathematical problem decomposition which is the main source of scalable parallelism. In this paper we describe a first approach towards the development of a fully scalable 4D-Var DA algorithm which is essentially based on the introduction of concurrency from beginning in the mathematical model. The objective of this paper is to prove the feasibility of a problem/domain decomposition of the 4D Var functional and to provide first experiences on its scalability on a test case based on Shallow Water Equations.

II. Preliminary Concepts

Let $t \in [0,T]$ and $x \in \Re^S$. Let f be a function belonging to the Hilbert space $\mathcal{K}([0,T] \times \Omega)$, that is:

$$f(t,x) : [0,T] \times \Omega \mapsto \mathcal{K}$$

Let $\Omega \subset \Re^S$ be decomposed into a sequence of p overlapping sub-domains $\Omega_i \subset \Re^{r_i}$, $r_i \leq N$, $i = 1, \ldots, p$ such that

$$\Omega = \bigcup_{i=1}^{p} \Omega_i \tag{1}$$

where

$$\Omega_i \cap \Omega_j = \Omega_{ij} \neq \emptyset$$

if the sub-domains are adjacent.
In the same way, let $[0,T]$ be decomposed into a sequence of r overlapping sub-intervals $[t_i, t_{i+1}]$ such that $t_0 = 0$ and $t_r = T$.

Associated with such decomposition, we define the Restriction Operator. Let us define:

$$RO_{ij} : \mathcal{K}([0,T] \times \Omega) \mapsto \mathcal{K}([t_i, t_{i+1}] \times \Omega_j)$$

such that:

$$RO_{ij}(f(t,x)) \equiv \quad f(t,x), \qquad (t,x) \in [t_i, t_{i+1}] \times \Omega_j$$

Finally, we pose:

$$f_{ij}^{RO}(t,x) \equiv RO_{ij}[f(t,x)]$$

978-1-4673-7813-0/15/$31.00 ©2015 IEEE

In line with this, given a set of $p \times r$ functions g_{ij}, $i = 1, \ldots, r$, $j = 1, \ldots, p$ each belonging to the Hilbert space $\mathcal{K}([t_i, t_{i+1}] \times \Omega_j)$, we define the Extension Operator. Let us define:

$$EO_{ij} : \mathcal{K}([t_i, t_{i+1}] \times \Omega_j) \mapsto \mathcal{K}([0, T] \times \Omega)$$

such that:

$$EO_{ij}(g_{ij}(t,x)) = \begin{cases} g_{ij}(t,x) & t \in [t_i, t_{i+1}], x \in \Omega_j \\ 0 & elsewhere \end{cases}$$

Moreover, for simplicity of notations, we pose:

$$g_{ij}^{EO}(t,x) \equiv EO_{ij}[g_{ij}(t,x)]$$

Remark 1: We observe that, for any function $f \in \mathcal{K}([0,T] \times \Omega)$, associated to the domain decomposition (1), it holds that

$$f(t,x) = \sum_{i,j} EO_{ij}\left[f_{ij}^{RO}(t,x)\right]. \tag{2}$$

Remark 2: We will use the same notation to denote the restriction/extension operators acting on vectors. If $\mathbf{w} = (w_i^j) \in \Re^{NP}$ and if $\Omega = \{1, 2, 3, \ldots, NP\}$, then

$$RO_{i,j}(\mathbf{w}) \equiv \mathbf{w}^{\mathbf{RO_{i,j}}} \equiv (w_i)_{i \in \Omega_i}, \quad \mathbf{w}^{\mathbf{RO_{i,j}}} \in \Re^{r_i}.$$

In the same way, if $\mathbf{z} = (z_i)_{i \in \Omega_i}$, it is

$$EO_{i,j}(\mathbf{z}) = \begin{cases} z_k & k \in \Omega_i \\ 0 & elsewhere \end{cases}$$

and $EO_{i,j}(\mathbf{z}) \equiv \mathbf{z}^{\mathbf{EO_{i,j}}} \in \Re^{NP}$.

Let $\mathbf{w} \in \Re^{NP}$, and let $\mathbf{C}(\mathbf{w})$ denote the matrix such that:

$$\mathbf{C}(\mathbf{w}) = \mathbf{w}\mathbf{w}^T$$

Associated to the domain decomposition (1), let $\mathbf{C}(\mathbf{w}) \in \Re^{NP \times NP}$ be the covariance matrix of a random vector $\mathbf{w} \in \Re^{NP}$, that is coefficient $c_{i,j}$ of $\mathbf{C}$ is $c_{i,j} = \sigma_{ij} \equiv Cov(w_i, w_j)$. Let $s, q < NP$, we define the restriction operator $RO_{s,q}$ onto $\mathbf{C}(\mathbf{w})$ as follows:

$$RO_{s,q} : \mathbf{C}(\mathbf{w}) \in \Re^{N \times N} \mapsto RO_{s,q}[\mathbf{C}(\mathbf{w})] := \\ = \mathbf{C}(\mathbf{w}^{\mathbf{RO_{s,q}}}) \in \Re^{s \times q}$$

In the following, for simplicity of notations, we refer to $\mathbf{C}(\mathbf{w}^{\mathbf{RO_{s,q}}})$ as $\mathbf{C_{s,q}}$.

III. The 4D-Var Data Assimilation Problem

The method that we are describing is called Four-Dimensional because it takes into account observations that are distributed in space and over an interval of time (typically 6 or 12 hours), often called the analysis window.
Let $t \in [0, T]$, i.e. the analysis window. Let assume that the evolution state $u(t,x)$ of a predictive system is governed by the mathematical model

$$\mathcal{M}: \quad u(t - \Delta t, x) \mapsto u(t,x), \qquad \forall\, (t,x) \in [0,T] \times \Omega$$

with $u(t_0, x)$, $t_0 = 0$ as initial condition. Let:

$$\mathcal{H}: \quad u(t,x) \mapsto v(t,x), \qquad \forall\, (t,x) \in [0,T] \times \Omega$$

denote the observations mapping, where $\mathcal{H}$ is a given nonlinear operator which includes transformations and grid interpolations.
According to the practical applications of model-based assimilation of observations, we will use the following definition of 4D Variational DA.
Given

- $D_{NP}(\Omega) = \{x_j\}_{j=1,\ldots,NP}$: discretisation of $\Omega \subset \Re^3$;
- $\{t_k\}_{k=0,1,\ldots,N-1}$: discretisation of $[0,T]$;
- $\mathbf{u}_0 = \{u_0^j\}_{j=1,\ldots,NP} \equiv \{u(t_0, x_j)\}_{j=1,\ldots,NP} \in \Re^{NP}$: the background estimates, i.e. the state at time t_0.
- for each $k = 0, \ldots, N-1$,
  - $\mathbf{M}(t_k, t_{k+1})$: discretisation of the first order approximation of $\mathcal{M}$ from time t_k to t_{k+1};
  - $\{u_k^j\}_{j=1,\ldots,NP} \equiv \{u(t_k, x_j)\}_{j=1,\ldots,NP} \in \Re^{NP}$: the vector of the numerical solution of $\mathbf{M}(t_{k-1}, t_k)$ at t_k;
  - $\{v(t_k, y_j)\}_{j=1,\ldots,nobs}$: the vector of the observations at t_k;
  - $\mathcal{H}(u(t,x)) \simeq \mathcal{H}(u(t_k, z_j)) + \mathbf{H}_k(u(t_k, x_j) - u(t_k, z_j))$: a linearization of $\mathcal{H}$, where $\mathbf{H}_k \in \Re^{NP \times nobs}$ is the matrix obtained by the discretisation of the Jacobian of $\mathcal{H}$ and $nobs << NP$;
  - $\mathbf{R_k}$ and $\mathbf{B_k}$: the diagonal covariance matrices of the errors on $\mathbf{v}_k$ and on $\mathbf{u}_k$, respectively. These matrices are symmetric and positive definite.

Def: [The DA inverse problem]
The DA problem can be mathematically described as the following inverse problem: given

1) the vector $\mathbf{v} = (\mathbf{v}_k)_k \in \Re^{N \times NP}$,
2) the block diagonal matrix

$$\mathbf{G} = [\mathbf{H}_0\mathbf{M}(t_0, t_1), \mathbf{H}_1\mathbf{M}(t_1, t_2), \ldots, \mathbf{H}_{N-1}\mathbf{M}(t_{N-1}, t_N)]$$

to compute $\mathbf{u}^{DA} = (\mathbf{u}_0^{DA}, \mathbf{u}_1^{DA}, \ldots, \mathbf{u}_{N-1}^{DA}) \in \Re^{NP \times N}$ (the so called analysis in $t_0, t_1, \ldots, t_{N-1}$ such that

$$\mathbf{v} = \mathbf{G}[\mathbf{u}^{DA}]$$

subject to the constraint that in t_0:

$$\mathbf{u}_0^{DA} = \mathbf{u}_0$$

Since $\mathbf{G}$ is typically rank deficient, DA is an ill posed inverse problem [8]. The Tikhonov-regularized formulation leads to

an unconstrained least square problem, where the weighted background term acts as a regularization term, which ensure the existence of a solution and also damp the sensitivity of the solution to the observational errors.

Variational DA problem can be described as following:

$$\begin{aligned} \mathbf{u}^{DA} &= argmin_{\mathbf{u}} J(\mathbf{u}) \\ &= argmin_u \left\{ \|\mathbf{G}\mathbf{u}-\mathbf{v}\|^2_{\mathbf{R}} + \|\mathbf{u}-\mathbf{u}_0\|^2_{\mathbf{B}} \right\} \end{aligned} \quad (3)$$

where $\mathbf{B} = diag(B_k)$, $\mathbf{R} = diag(R_k)$, while $\|\cdot\|_{\mathbf{B}}$ and $\|\cdot\|_{\mathbf{R}}$ denote the weighted euclidean norm.

So, the 4D-VAR DA inverse problem leads to the minimization of the functional:

$$J(\mathbf{u}) = \sum_{k=0}^{N-1} (\mathbf{G}_k\mathbf{u}-\mathbf{v})^T\mathbf{R}(\mathbf{G}_k\mathbf{u}-\mathbf{v}) + (\mathbf{u}-\mathbf{u}_0)^T\mathbf{B}(\mathbf{u}-\mathbf{u}_0) \quad (4)$$

where

$$\mathbf{G}_k = \mathbf{H}_k\mathbf{M}(t_k, t_{k+1}), \qquad k = 0, \ldots, N-1$$

It depends on $\mathbf{u}$, $\mathbf{v}$, $\mathbf{R}$, $\mathbf{B}$, Ω, and $[t_0, T_N]$, then to emphasize these relationship, we write $J(\mathbf{u}, \mathbf{v}, \mathbf{R}, \mathbf{B}, \Omega, [t_0, t_N])$.

Remark 3: We observe that instead of (4) we will employ the preconditioned 4D Var functional where the preconditioner is obtained by using the Cholesky factorization of $\mathbf{B} = VV^T$ we already described in [3] applied to a 3DVar model. Therefore, the preconditioned 4D Var functional is

$$J(\mathbf{w}) = \|\mathbf{w}\|_2 + \sum_{k=0}^{N-1} \|\mathbf{G}_k V\mathbf{w} - \mathbf{d}\|_{\mathbf{R}} \quad (5)$$

where

$$\mathbf{u} = \mathbf{u}_0 + \delta\mathbf{u}, \quad \mathbf{w} = V^T\delta\mathbf{u}, \quad \mathbf{d} = \mathbf{v} - \mathbf{G}\mathbf{u}$$

As the minimum of J in (5) is computed by using a quasi Newton method (the L-BFGS method), we need to compute ∇J which is given in (6):

$$\begin{aligned} &\nabla J(\mathbf{w}) = \mathbf{w} + \\ &+\frac{1}{2}\sum_{k=0}^{N-1} (V\widetilde{\mathbf{M}}^*(t_0, t_k)\mathbf{H}_k)^T\mathbf{R}^{-1}(\mathbf{H}_k\widetilde{\mathbf{M}}^*(t_0, t_k)V\mathbf{w}_k - d_k) \end{aligned} \quad (6)$$

where $\widetilde{\mathbf{M}}^*(t_0, t_k)$ is the adjoint operator of the tangent linear approximation of

$$\mathbf{M}(t_0, t_k) = \mathbf{M}(t_0, t_1) \cdot \mathbf{M}(t_1, t_2) \cdot \ldots \cdot \mathbf{M}(t_{k-1}, t_k) \quad (7)$$

The relation (7) means that the model evolution in $[t_0, t_k]$ is obtained by combining the evolution states along the previous k time sub intervals.

IV. The Problem Decomposition

In this work, in order to validate our approach, we focus on the decomposition of the spatial domain; more precisely, we decompose the domain $[t_0, t_N]\times\Omega$, in sub-domains $[t_0, t_N]\times\Omega_i$. Therefore, the restriction operator RO_i refers to function and functional restriction on $[t_0, t_N] \times \Omega_i$.
In order to solve the mathematical problem on the subdomains, we have to define the *local* 4D-VAR regularization functionals J_i. The local 4D-VAR regularization functional, which describes the local DA problems on each sub-domain $[t_0, t_N]\times\Omega_i$ is defined in (8). It is obtained by first applying the restriction operator to the 4D-VAR regularization functional J, then by adding a *local* constraint to such restriction. This is in order to enforce the continuity of each solution of the local DA problem onto the overlap region between adjacent domains:

$$\begin{aligned} J_i = J(\mathbf{u}^{\mathbf{RO_i}}, \mathbf{v}^{\mathbf{RO_i}}, \mathbf{R_i}, \mathbf{B_i}, \Omega_i, [t_0, t_N]) + \\ +\|\mathbf{M}(t_0, t_{N-1})V\mathbf{w}^+ - \mathbf{M}(t_0, t_{N-1})V\mathbf{w}^-\|_2 \end{aligned} \quad (8)$$

where $\mathbf{w}^+$ and $\mathbf{w}^-$ denote the values of the vector $\mathbf{w}$ on the overlapping region (domain) between two adjacent domains, on the left side and the right side of the overlap, respectively.

V. The Scalable Parallel Algoritm

Let $\mathcal{A}_{M4D}$ denote the algorithm which solves the local 4D-VAR problems on each subdomain $[t_0, t_{N-1}] \times \Omega_j$.

$\mathcal{A}_{M4D}(NP, p)$: Modified 4D-VAR algorithm on $[t_0, t_{N-1}] \times \Omega_i$

1: **Input:** $\mathbf{v}_i$ and $\mathbf{u}^{\mathbf{RO_i}}$
2: **Define** $\mathbf{H_i}$
3: **Compute** $\mathbf{d_i} \leftarrow \mathbf{v}_i - \mathbf{H_i}\mathbf{u}^{\mathbf{RO_i}}$
4: **Define** $\mathbf{R_i}$ and $\mathbf{B_i}$
5: **Compute** the matrix $\mathbf{V_i}$ from $\mathbf{B_i}$
6: **Define** the initial value of $\mathbf{u}^{DA_i}$
7: **Compute** $\mathbf{w}_i \leftarrow \mathbf{V_i}^T\mathbf{u}^{DA_i}$
8: **repeat**
9: **Send and Receive** the boundary conditions from the adjacent domains
10: **Compute** $\mathbf{J_i} \leftarrow \mathbf{J_i}(\mathbf{w_i})$
11: **Compute** $\mathbf{gradJ_i} \leftarrow \nabla\mathbf{J_i}(\mathbf{w_i})$
12: **Compute** new values for $\mathbf{w}_i$ by the L-BFGS steps
13: **until** (Convergence on $\mathbf{w_i}$ is obtained)
14: **Compute** $\mathbf{u}_i^{DA} \leftarrow (\mathbf{u_0})^{\mathbf{RO_i}} + \mathbf{V_i}\mathbf{w_i}$

The convergence criterion used in the repeat-until loop, at each iteration $\mathbf{iter}$, is

$$\frac{\mathbf{J_i(iter+1)} - \mathbf{J_i(iter)})}{\max\{|\mathbf{J_i(iter)}|, |\mathbf{J_i(iter+1)}|, 1\}} \leq TOL$$

where $TOL = 10^{-6}$ together to a maximum iteration number (which is about ten in DA applications).

Observe that $\mathcal{A}_{M4D}(NP,p)$ just requires an exchange of boundary conditions (local data communications) between adjacent sub-domains (which produces the surface-to-volume effect) [2].

VI. Experiments

In order to validate the proposed approach we described some results related to the quality of the numerical results and in terms of reduction in computation time. All considerations made on a case study based on the linear Shallow Water Equation (SWE) [11] as a simplified version of a forecasting one: for that reason the shallow water model can be considered a consistent tool to get a proof of concept of the validity of the approach. For the the linear SWE the adjoint operator is the transpose of the linear model [14].

The SWE is spatially discretised with a centered finite-difference scheme in space and an explicit leapfrog integration scheme in time.
We assume that $NP = n_x \times n_y \times n_z$ and $n_x = n_y = n$ while $n_z = 1$ Since the unknown vectors are 3 (the fluid height or depth, and the two-dimensional fluid velocity fields), the problem size is $NP = 3n^2$.

Let $\Omega = [0, n+1] \times [0, n+1] \times 1$ and $D_{NP}(\Omega) = \{x_j\}_{j=1,\ldots,NP}$ where $x_j = j$. Let $[0, \infty[$, be the time interval and $t_k = k\Delta t$ where $\Delta t = 0.01$, which is also the step size for time integration of the shallow water model.

$\mathbf{H}$ is assumed to be a piecewise linear interpolation operator whose coefficients are computed using the points of model domain nearest the observation values. Matrix $\mathbf{R}$ is

$$\mathbf{R} = \sigma_o^2 \mathbf{I}, \quad \sigma_o^2 = 0.5$$

Observation values are randomly chosen among the values of $\mathbf{u}^{\mathcal{M}}$.

From the modeler's point of view, the algorithm running on each processor/sub domain is almost identical to the "mono-processor/domain" code.

A. Performance analysis

Let $nproc = s \times q$. We assume a 2D uniform decomposition of $D_{NP}(\Omega)$ along the (x,y)-axis, which is the x-axis is divided by s and the y-axis by q. Implementation of $\mathcal{A}_{M4D}(NP,p)$ was obtained mapping each sub-domain onto a processing element of the reference parallel architecture, i.e. we use the following correspondence

$$p \leftrightarrow nproc$$

The size of each sub-domain Ω_i is

$$r_i = \frac{NP}{nproc} = nloc_x \times nloc_y \times nloc_z + 2o_x \times 2o_y$$

where:

$$nloc_x = \frac{n_x}{s} + 2o_x \ , \ nloc_y = \frac{n_y}{q} + 2o_y \ , \ nloc_z = n_z \ . \quad (9)$$

These dimensions include the overlapping $(2o_x \times 2o_y)$.

By using n_{xpp} and n_{ypp} to denote the position of the $(1,1)$ grid-point of each sub domain in the global domain, each element of the local array x^{loc} corresponds to the element of the global array x^{glob}, as (10):

$$x^{glob}(i + n_{xpp} - 1; j + n_{ypp} - 1; k) = x^{loc}(i; j; k) \quad (10)$$

where $1 < i < nloc_x$, $1 < j < nloc_y$, and $1 < k < nloc_z$.

Concerning the observations, we distribute them according to the "model distribution": sub-domains are associated with different geographic regions. We carry out a check on the geographical location of the observed data to attribute the observations to the processor related to the geographical region in which the measurement was made.

Let us consider the algorithm $\mathcal{A}_{M4D}(NP,p)$ running on a parallel architecture made of 8 distributed memory blades each one with 8 computing elements sharing the same local memory. In the numerical experiments we refer to them as processing elements ($nproc$).

Table I shows $\|\mathbf{u}^{true} - \mathbf{u}^{DA}\|_\infty$, where $\mathbf{u}^{DA}$ is computed as the sum of the local solutions $\widetilde{\mathbf{u}}^{DA_i}$ on Ω_i, and

$$\widetilde{\mathbf{u}}^{DA_i} = (\mathbf{u}^{\mathcal{M}})^{\mathbf{RO_i}} + \mathbf{V}_i \mathbf{w}_i.$$

TABLE I: Values of $\|\mathbf{u}^{true} - \mathbf{u}^{DA}\|_\infty$, for different values of $nproc$ and of n.

n	$nproc$	$\|\mathbf{u}^{true} - \mathbf{u}_{DA}\|_\infty$
$O(10^7)$	1	5.3264971853605180 8293e-03
	2	5.3286538578212611 0836e-03
	4	5.3286538578212333 5279e-03
	8	5.3129596505877424 4394e-03
	10	5.3133065672577584 0599e-03
$O(10^8)$	1	5.2842122783119810 3697e-03
	2	5.2842122783119810 3697e-03
	4	5.2842122783119810 3697e-03
	8	5.2515087080743372 2831e-03
	10	5.2842117708644803 5087e-03

To analyse the algorithm scalability we use the following performance metric [5]:

$$S_{p,1}^{S4D}(NP) = \frac{T(NP)}{pT(NP/p)} \quad . \quad (11)$$

where $T(NP)$ denotes the time complexity of the algorithm for a dimension problem NP.

Remark 4: Let t_{flop} denote the unitary time required by one floating point operation. As a result, the execution time needed to algorithm $\mathcal{A}_{4D}(NP)$ (or $\mathcal{A}_{M4D}(NP,1)$) for performing $T(N)$ floating point operations, is

$$T_{flop}(N) = T(N) \times t_{flop} \quad .$$

TABLE II: Execution time and *scale-up* factor for different values of $NP = 3n^2$ and $nproc$. The value for N is $N = 4$.

n	$nproc$	$T^{nproc}(NP)$	measured $S^{S4D}_{nproc,1}$	$S^{S4D}_{nproc,1}$
$O(10^8)$	1	1.6192e+03	1.0	1.00
	2	3.3286e+02	4.86	4.52
	4	1.0168e+02	15.92	29.71
	8	3.5930e+01	45.06	104.00

Multiplying and dividing the by t_{flop} the (11) becomes

$$S^{S4D}_{p,1}(NP) = \frac{T_{flop}(NP)}{pT_{flop}(NP/p)} \quad . \tag{12}$$

Since the time complexity of algorithm $\mathcal{A}_{4D}(NP)$ is $T(NP) = O(It\dot{f}(NP))$ flops, on a problem of size NP, where $f(NP) \in \Pi_2$ where It is a constant measuring the (maximum) iteration number of the iterative method used for computing the Truncated Singular Value Decomposition and the minimum of the gradient of the functional J, following result specifies the scale-up factor of algorithm $\mathcal{A}_{S4D}(NP, p)$:

$$S^{S4D}_{p,1}(NP) = O(It\, p^3) \quad . \tag{13}$$

If $T^{nproc}(NP)$ denotes the execution time of $\mathcal{A}_{S4D}(NP, p)$, measured in seconds, it is

$$T^{nproc}(NP) = T^{nproc}_{flop}(NP) + T^{nproc}_{com}(NP)$$

where

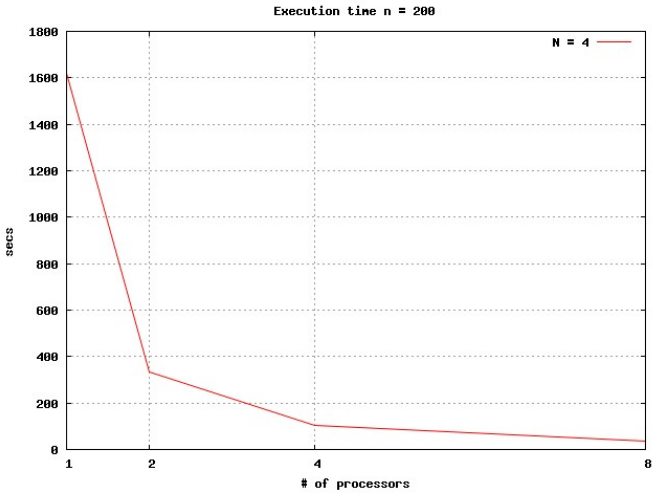
- $T^{nproc}_{flop}(NP)$ is computing time required for the execution of $T(NP)$ floating point operations;
- $T^{nproc}_{oh}(NP)$ is inter-node communication time of $T(NP)$ data.

We report $T^{nproc}(NP)$ [1] and measured values of S^{S4D}_{nproc,p_1} compared to those of S^{S4D}_{nproc,p_1} as given in the (13), where $p_1 = 1$, $p_2 = nproc$ and $p = \frac{p_2}{p_1}$.

VII. Discussion

In order to discuss results in Tables 1 and 2 in the following we report a brief discussion by considering, for simplicity of notations, the measured value of the scale up factor $S_{nproc,1}$. The algorithms can now be analyzed with simplified models, i.e. we disregard memory management, pipelining, message routing, synchronization, etc., and concentrate only on the arithmetic work (load balance) and the interprocessor communication. We estimate the CPU-time by counting the operations and the processor to processor messages. In the case of local operations we take the CPU-time of the most loaded processor.

[1]Execution times are collected on a cluster of nodes, connected by 10 Gigabit Ethernet technology, each of them with processors Intel Xeon E5-4610v2@2.30GHz.



(a)

Trends of execution times (a) as function of the $nproc$ values.

Let

$$s_{nproc} = \frac{T_{flop}(NP/p)}{T^{nproc}_{flops}(NP/p)}$$

denote the speed up of the algorithm running on $nproc$ processing elements for solving the local DA problem, then we get:

$$\text{measured} \quad S_{nproc,1} = \frac{T_{flop}(NP)}{\frac{pT_{flop}(NP/p)}{s_{nproc}} + pT_{oh}(NP/p)} = \\ = \frac{s_{nproc}\frac{T_{flop}(NP)}{pT_{flop}(NP/p)}}{1 + \frac{s_{nproc}T_{oh}(NP/p)}{T_{flop}(NP/p)}}. \tag{14}$$

As we need to guarantee that the so-called *surface-to-volume* effect on each local DA problem is produced [2], [6], we assume:

$$0 \leq \frac{S}{V} := \frac{T_{oh}(N/p)}{T_{flop}(N/p)} < 1 - \frac{1}{s^{loc}_{nproc}} < 1 \quad .$$

If

$$\alpha := \frac{s^{loc}_{nproc}}{1 + \frac{s^{loc}_{nproc}T_{oh}(N/p)}{T_{flop}(N/p)}} = \frac{s^{loc}_{nproc}}{1 + s^{loc}_{nproc}\frac{S}{V}}$$

from (14) it comes out that

$$S^{measured}_{1,nproc} = \alpha S_{1,nproc}.$$

It holds that

a if $s^{loc}_{nproc} = 1$ then

$$\alpha < 1 \Leftrightarrow S^{measured}_{nproc,1} < S_{nproc,1}$$

b if $s^{loc}_{nproc} > 1$ then

$$\alpha > 1 \Leftrightarrow S^{measured}_{nproc,1} > S_{nproc,1};$$

c if $s^{loc}_{nproc} = p$ then

$$1 < \alpha < p \Rightarrow S^{measured}_{nproc,1} < pS_{nproc,1};$$

Hence, we may conclude that if

$$s^{loc}_{nproc} \in]1, p] \Rightarrow S^{measured}_{nproc} \in]S_{nproc,1}, p\, S_{nproc,1}[.$$

Since in our experiments it is $s_{nproc} = 1$, this analysis validates the experimental results.

Concerning the data distribution, there are three strategies for their decomposition. The first alternative is to distribute the observations according to the distribution in the model. This data distribution is called the model distribution. The drawback is the lack of load balance anyway no extra communication is needed. In the second alternative, all data are distributed geographically in connected domains, balancing the work load in the procedure for transformation to observed quantities. This is the geographic distribution. Now we have a good load balance but must do costly re-distributions of the grid-point variables. Such strategy, with equal work load on connected domains, seems to be preferable on shared memory machines. Finally, in the third alternative two distributions are handled at the same time. In this strategy the work load is balanced with a minimal amount of communication. Here, since the aim is to analyse the inherent algorithmic scalability, we employ the model distribution strategy which does not require extra communication overhead. As expected, we may observe the impact of the computational workload imbalance on the measured scaleup as the processor number grows.

VIII. Conclusion

We proposed a Problem Decomposition approach to Variational Data Assimilation. The novel approach consists on a new formulation of the 4D-Var DA variational scheme on each subdomain. Our idea is to decompose the 4D-Var DA functional according to the physical domain decomposition obtaining a set of DA-subproblems, each one defined in a sub domain, to be solved concurrently. We use a case study based on the shallow water model to build a simulated case (a sort of phantom experiment) to verify the correctness of the solution.

Acknowledgment

This work has been realised thanks to the use of the SCoPE computing infrastructure at the University of Naples Federico II.

References

[1] R. Arcucci, L. D'Amore, S. Celestino, G. Scotti, G.Laccetti, A Parallel Approach for 3D-Variational Data Assimilation on GPUs in Ocean Circulation Models, International Conference on Scientific Computing, World Academy of Science, Engineering and Technology Proceeding, Vol. 2(5), ISSN 1307-6892, Paris, 2015.

[2] L. Carracciuolo, D'Amore L., Murli, A., Towards a parallel component for imaging in PETSc programming environment: A case study in 3-D echocardiography, Parallel Computing, Volume 32, Issue 1, January 2006, pp. 67-83.

[3] L. D'Amore, R. Arcucci, L. Carracciuolo, A. Murli, A Scalable Approach to Variational Data Assimilation, Journal of Scientific Computing, Vol. 61, 2014

[4] L. D'Amore, R. Arcucci, L. Carracciuolo, A. Murli -DD-OceanVar: a Domain Decomposition fully parallel Data Assimilation software in Mediterranean Sea - Procedia Computer Science 18, 2013, pp. 1235-1244.

[5] L. D'Amore, R. Arcucci, L. Marcellino and A. Murli, HPC computation issues of the incremental 3D variational data assimilation scheme in OceanVar software - Journal of Numerical Analysis, Industrial and Applied Mathematics vol. 7, no. 3-4, 2012, pp. 91-105

[6] L. D'Amore, D. Casaburi, A. Galletti, L. Marcellino, A. Murli - Integration of emerging computer technologies for an efficient image sequences analysis, Integrated Computer-Aided Engineering, vol. 18, Issue 4, 2011, pp. 365-378.

[7] M. Fisher, Parallelisation in the time dimension of 4D-Var, ECMWF, December 2014

[8] S.A. Haben, A.S. Lawless, N.K. Nichols: Conditioning of the 3DVAR Data Assimilation Problem, Mathematics Report 3/2009. Department of Mathematics, University of Reading, 2009.

[9] H. P. Flatt, K. Kennedy - Performance of Parallel Processors, Parallel Computing, Vol. 12, 1989, pp. 1–20.

[10] E. Kalnay - Atmospheric Modeling, Data Assimilation and Predictability. - Cambridge University Press, Cambridge, MA, 2003.

[11] C. Moler. Experiments with MATLAB, 2011.

[12] A. M. Moore, H. G. Arango, G. Broquet, B. S. Powell, A. T. Weaver, J. Zavala-Garay, The Regional Ocean Modeling System (ROMS) 4-dimensional variational data assimilation systems Part I System overview and formulation, Progress in Oceanography, 91 (2011) pp. 3449.

[13] A. M. Moore, H. G. Arango, G. Broquet, B. S. Powell, A. T. Weaver, J. Zavala-Garay, The Regional Ocean Modeling System (ROMS) 4D-Var Assimilation Systems applied to the California Current System, 2014

[14] K.Y. Wang, D.J. Lary, D.E. Shallcross, S.M. Hall, J.A. Pyle, A review on the use of the adjoint method in four-dimensional athmospheric-chemistry data assimilation. Q.J.R.Metereol.Soc., 2001, 127, pp.2181-2204.

[15] C. Zhu, R.H. Byrd, P. Lu, and J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization, ACM Trans. Math. Softw. 23, 1997, pp. 550-560

[16] Xin Zhang, Xiang-Yu Huang, Jianyu Liu, WRF Four-dimensional variational data assimilation system Tutorial for V3.6, WRFDA Tutorial, 2014