Scalability of Data Assimilation at ECMWF

Yannick Trémolet

ECMWF

30 October 2014
Our current 4D-Var implementation is not scalable enough for the future.
Improving 4D-Var Scalability

- Data assimilation specific code optimizations:
  - Reduce input observations to the ones actually assimilated (COPE),
  - Reduce I/O (linearization state, preconditioner, ODB): single executable.

- The forecast model is one important component of the data assimilation system (80% of runtime): improvements in the model (and TL/AD) will benefit data assimilation directly.

- The same code adaptations as in the model will be used in other parts of the system (observation operators, covariance matrices, ...)

- However, the lower resolution of most of the data assimilation system (≈20 times less grid-points) makes scalability more problematic.
  - Sequential nature of the minimization algorithm.
Forecast models are very sequential by nature.

- Scalability becomes worse at high resolution because time steps get shorter.
- Each time step must run faster: strong scalability is needed!
4D-Var doesn’t scale: is it that bad?

- Forecast models are very sequential by nature.
  - Scalability becomes worse at high resolution because time steps get shorter.
  - Each time step must run faster: strong scalability is needed!

- The number of iterations in 4D-Var is independent of resolution.
4D-Var doesn’t scale: is it that bad?

- Forecast models are very sequential by nature.
  - Scalability becomes worse at high resolution because time steps get shorter.
  - Each time step must run faster: strong scalability is needed!

- The number of iterations in 4D-Var is independent of resolution.
  - The number of iterations in 4D-Var has not changed since the 1990's
  - We only need the time per iteration to stay constant: weak scalability is enough!
4D-Var doesn’t scale: is it that bad?

- Forecast models are very sequential by nature.
  - Scalability becomes worse at high resolution because time steps get shorter.
  - Each time step must run faster: strong scalability is needed!

- The number of iterations in 4D-Var is independent of resolution.
  - The number of iterations in 4D-Var has not changed since the 1990's
  - We only need the time per iteration to stay constant: weak scalability is enough!

- We get it from the model: as long as the forecast model delivers the same forecast days per day, 4D-Var will run in the same wall clock time.
4D-Var doesn’t scale: is it that bad?

- Forecast models are very sequential by nature.
  - Scalability becomes worse at high resolution because time steps get shorter.
  - Each time step must run faster: strong scalability is needed!

- The number of iterations in 4D-Var is independent of resolution.
  - The number of iterations in 4D-Var has not changed since the 1990's
  - We only need the time per iteration to stay constant: weak scalability is enough!

- We get it from the model: as long as the forecast model delivers the same forecast days per day, 4D-Var will run in the same wall clock time.

- Do we need more scalability?
  - Yes, if we want to do better than 4D-Var (better science, better analysis, better forecasts...)

Y. Trémolet
Data Assimilation Scalability
3 / 14
Hybrid Data Assimilation

- Today’s best data assimilation algorithms are hybrid.
  - Ensemble DA system for computing background error covariances,
  - Variational DA system to provide the high resolution analysis.

- Hybrid data assimilation systems are very complex.

- There is a choice of ensemble DA methods to choose from.

- 4D-Var provides the best high resolution analysis:
  - one trend is to remove the variational component,
  - another is to improve it (scientifically) and make it more scalable.
ECMWF uses an ensemble of 4D-Vars to estimate background error statistics.

An alternative: EnKF
- Approximation of a Kalman filter with covariances projected on ensemble space, with issues related to localisation and inflation.
- A research implementation is maintained at ECMWF.

An alternative: 4D-En-Var
- Approximation of 4D-Var where time evolution of increments and covariances are projected on ensemble space (with localization),
- Available in OOPS.

All ensemble DA algorithms require running the members and combining information:
- The ensemble part scale with the number of members.
- The combining of information is cheap in computing but expensive in memory and/or I/O. Is that good for the future?
- The overall cost varies but so far scalability is good.
Ensemble Data Assimilation

- ECMWF uses an ensemble of 4D-Vars to estimate background error statistics.

- An alternative: EnKF
  - Approximation of a Kalman filter with covariances projected on ensemble space, with issues related to localisation and inflation.
  - A research implementation is maintained at ECMWF.

- An alternative: 4D-En-Var
  - Approximation of 4D-Var where time evolution of increments and covariances are projected on ensemble space (with localization),
  - Available in OOPS.

- All ensemble DA algorithms require running the members and combining information:
  - The ensemble part scale with the number of members.
  - The combining of information is cheap in computing but expensive in memory and/or I/O. Is that good for the future?
  - The overall cost varies but so far scalability is good.
Parallelism in 4D-Var

- Some aspects of data assimilation that are not available in the forecast model can be exploited.
Parallelism in 4D-Var

- Some aspects of data assimilation that are not available in the forecast model can be exploited.

The observations and background state are available throughout the whole window when we start the assimilation: it is possible to envisage parallelism in the time dimension.

- Assimilation should be considered a 4D problem, not an initial value problem.
Weak Constraint 4D-Var

- The control variable is 4D, with some flexibility w.r.t. the resolution in time (it is already the case in the spatial dimensions).

- Model integrations within each time-step (or sub-window) are independent.
  - $M$ and $H$ can run in parallel for each time-step or sub-window.
Weak Constraint 4D-Var

- The control variable is 4D, with some flexibility w.r.t. the resolution in time (it is already the case in the spatial dimensions).

- Model integrations within each time-step (or sub-window) are independent.
  - $\mathcal{M}$ and $\mathcal{H}$ can run in parallel for each time-step or sub-window.

- The additional model error terms ($J_q$) make the minimization and preconditioning more complex: we need to explore dual (i.e. observation) or mixed primal/dual space algorithms.

- It is also a theoretical improvement of 4D-Var (accounting for model error).
Saddle-Point Formulation of 4D-Var

- The weak constraint 4D-Var problem can be written as a constrained minimisation problem.

- The inner loop minimization problem is replaced by a saddle point optimization problem (Lagrange multiplier approach).

A few interesting properties are:
- The inverses of the covariance matrices are not needed,
- The parallelism over sub-windows is preserved,
- The tangent linear and adjoint models can run in parallel.
The saddle point formulation of 4D-Var is more scalable.

Weak constraint 4D-Var is also theoretically better than strong constraint 4D-Var although some questions remain (model error covariance matrix).
Object-Oriented Programming

- Exploring parallelism in new directions, through hybrid methods, weak constraint 4D-Var, new minimization algorithms or other techniques, requires considerable changes in the high level data assimilation algorithm.

- All that while getting ready for potential dramatic changes in the model...

- We need a very flexible, reliable, efficient, readable and modular code.
  - Readability improves staff efficiency: it is as important as computational efficiency (it’s just more difficult to measure).
  - Modularity improves staff scalability: it is as important as computational scalability (it’s just more difficult to measure).

- This is not specific to the IFS: the techniques that have emerged in the software industry to answer these needs are called **generic** and **object-oriented** programming.
The high levels Applications use abstract building blocks.

The Models implement the building blocks.

OOPS is independent of the Model being driven.
From IFS to OOPS

- The main idea is to keep the computational parts of the existing code and reuse them in a re-designed flexible structure.

- This can be achieved by a top-down and bottom-up approach.
  - From the top: Develop a new, modern, flexible structure (C++).
  - From the bottom: Progressively create self-contained units of code (Fortran).
  - Put the two together: Extract self-contained parts of the IFS and plug them into OOPS.

- From a Fortran point of view, this implies:
  - No global variables,
  - Control via interfaces (derived types passed by arguments).

- This is done at high level in the code.
  - It complements work on code optimisation done at lower level.
OOPS Benefits

- Code components are independent:
  - Components can easily be developed in parallel.
  - Their complexity decreases: less bugs and easier testing and debugging.

- Improved flexibility:
  - Develop new data assimilation (and other) science.
  - Explore and improve scalability.
  - Changes in one application do not affect other applications.
  - Ability to handle different models opens the door for coupled DA.

- OOPS does not solve scientific problems in itself: it provides a more powerful way to “tell the computer what to do”.
  - The OO layer developed for the simple models is not only a proof of concept: the same code is re-used to drive the IFS (generic).
Summary

4D-Var doesn’t scale well at the moment.

It is not as bad as it seems.

There are lots of interesting scientific directions to explore to make data assimilation better and more scalable!