Scalable Sequential Data Assimilation with the Parallel Data Assimilation Framework PDAF

Lars Nerger[1,2], Wolfgang Hiller[1], Jens Schröter[1]

(1) Alfred Wegener Institute for Polar and Marine Research
Bremerhaven, Germany

(2) Bremen Supercomputing Competence Center BremHLR

lars.nerger@awi.de

BremHLR
Kompetenzzentrum für Höchstleistungsrechnen Bremen

AWI

Overview

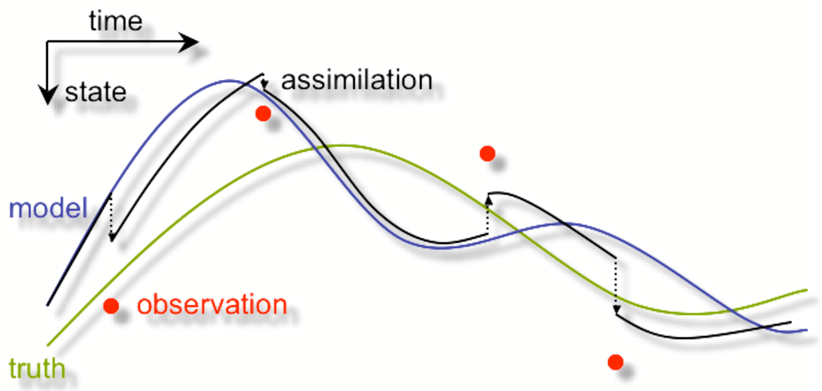
Focus on computational aspects of data assimilation

- Sequential data assimilation
- Parallel Data Assimilation Framework PDAF
- Parallel performance with PDAF

Sequential Data Assimilation

Goal

Combine model and observations for improved state representation



Method

Iteration:

Analysis:
Correct model state estimate when observations are available.

⟷

Forecast:
Propagate state and error estimate

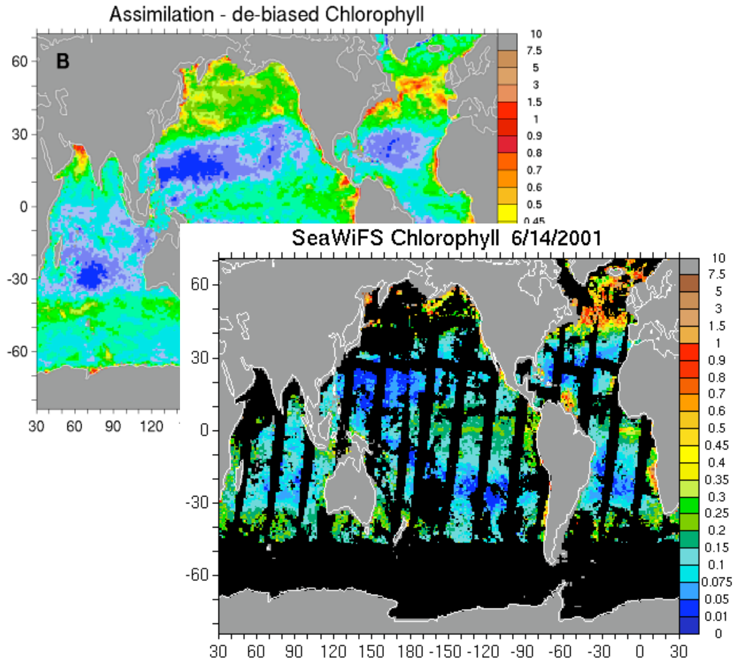
Common sequential algorithms

- Ensemble-based Kalman filters
- Particle filters

Application examples

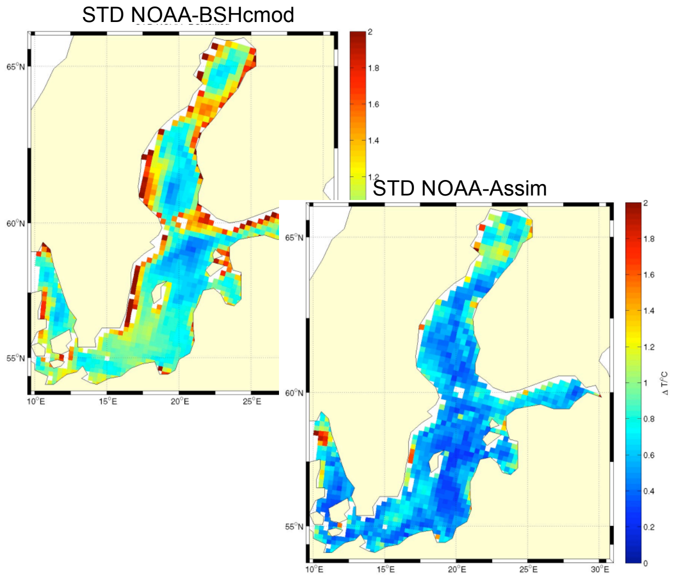
Ocean chlorophyll assimilation into NASA Ocean Biogeochemical Model (with Watson Gregg, NASA GSFC)

- ➢ Generation of daily re-analysis maps of chlorophyll at ocean surface
- ➢ Work toward multivariate assimilation



Coastal assimilation of ocean surface temperature (project "DeMarine Environment", AWI and BSH)

- ➢ North Sea and Baltic Sea
- ➢ Improve operational forecast skill, e.g. for storm surges



Computational and Practical Issues

Memory: Huge amount of memory required
(model fields and ensemble matrix)

Computing: Huge requirement of computing time
(ensemble integrations)

Parallelism: Natural parallelism of ensemble integration exists
- but needs to be implemented

Implementation: Existing models often not prepared for data assimilation

„Fixes“: Filter algorithms need „fixes“ and tuning
(literature provides typical methods)

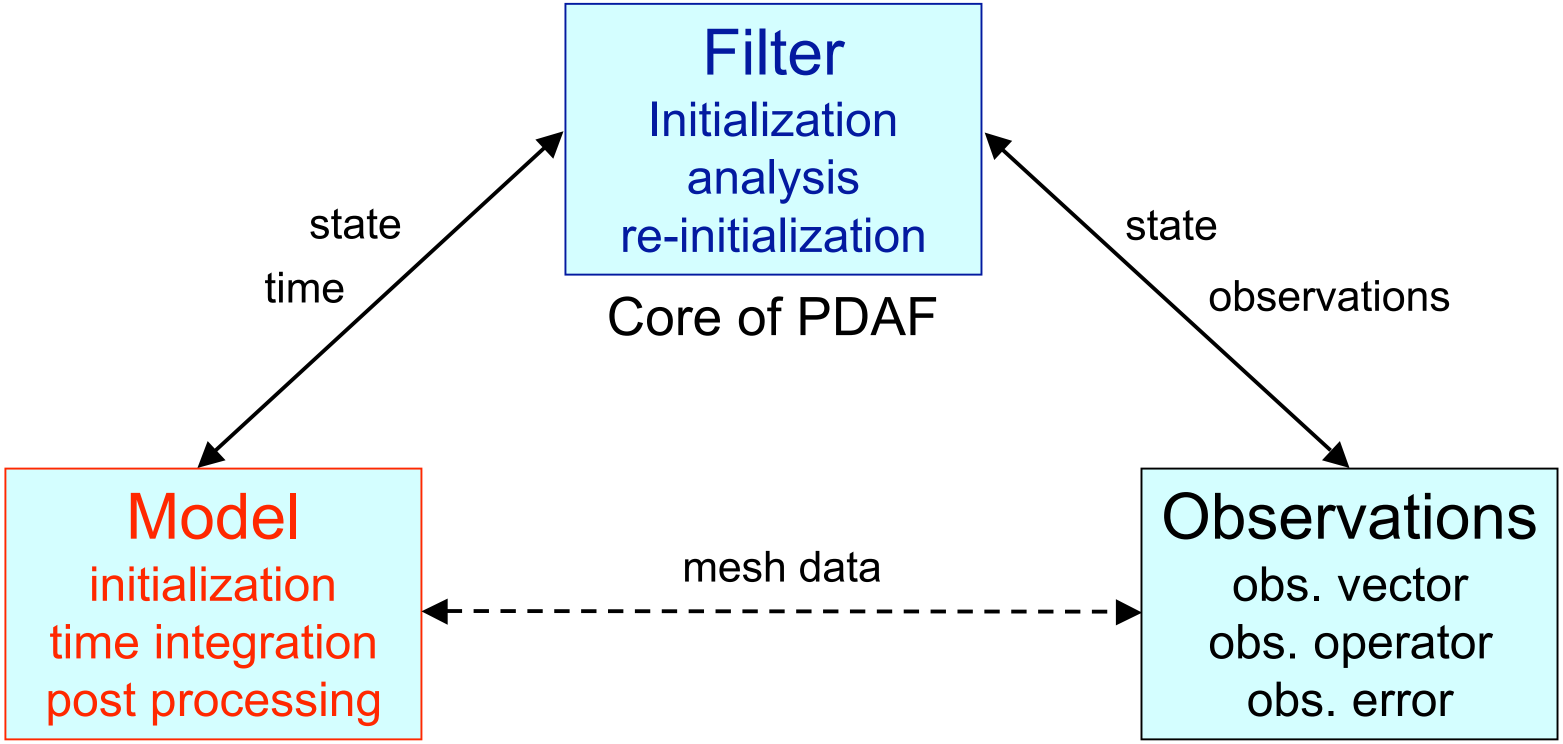
Motivation for a Framework

- Filter algorithms can be developed and implemented independently from model
- Parallelization of ensemble forecast can be implemented independently from model

A framework allows to

- Simplify implementation of data assimilation systems based on existing models
- Provide parallelization support for ensemble forecasts
- Provide parallelized and optimized filter algorithms
- Provide collection of „fixes“, which showed good performance in studies

Logical separation of assimilation system



⟷ Explicit interface

⇠- - -⇢ Exchange through module/common

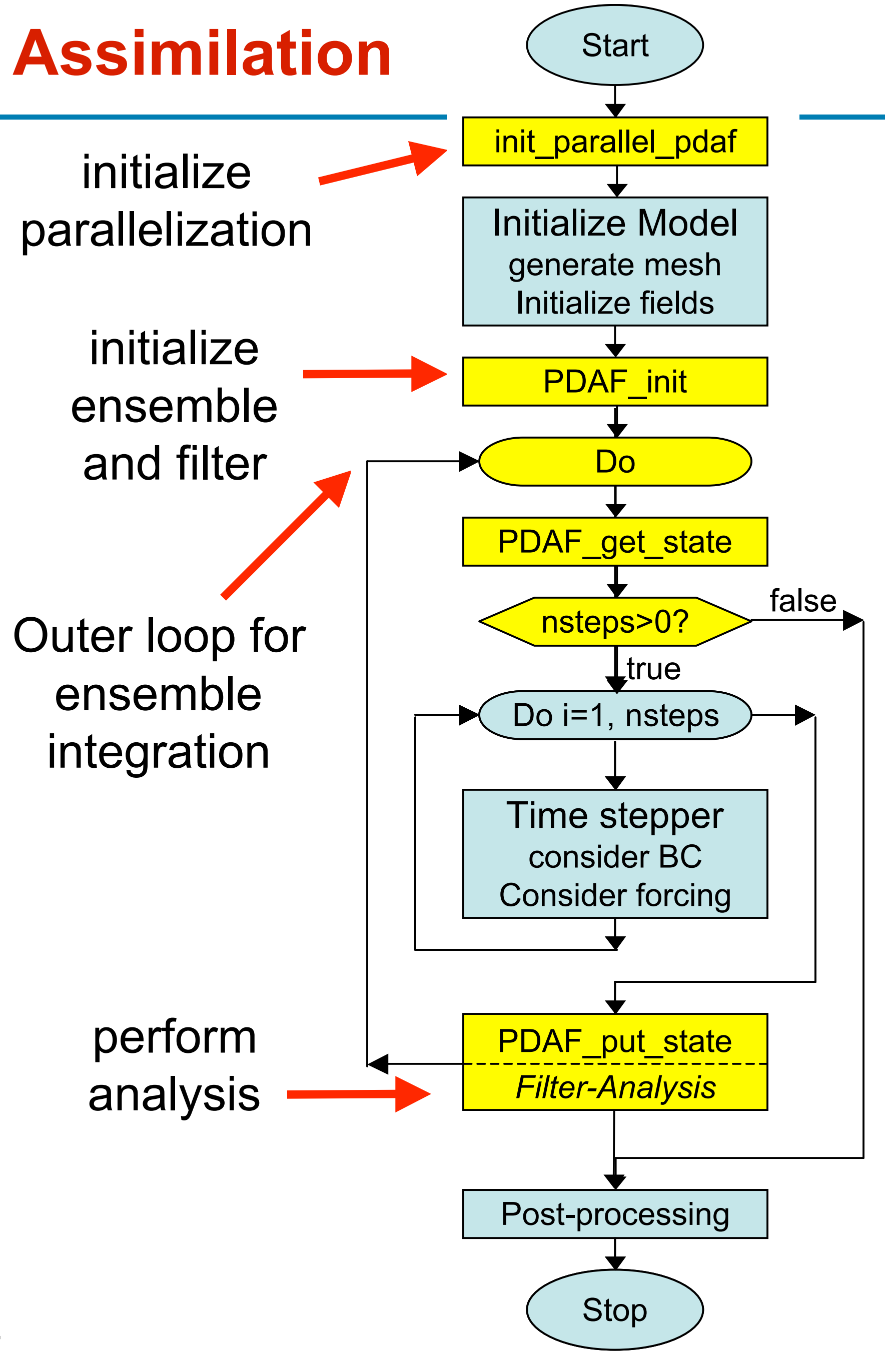
Extending a Model for Data Assimilation

Model

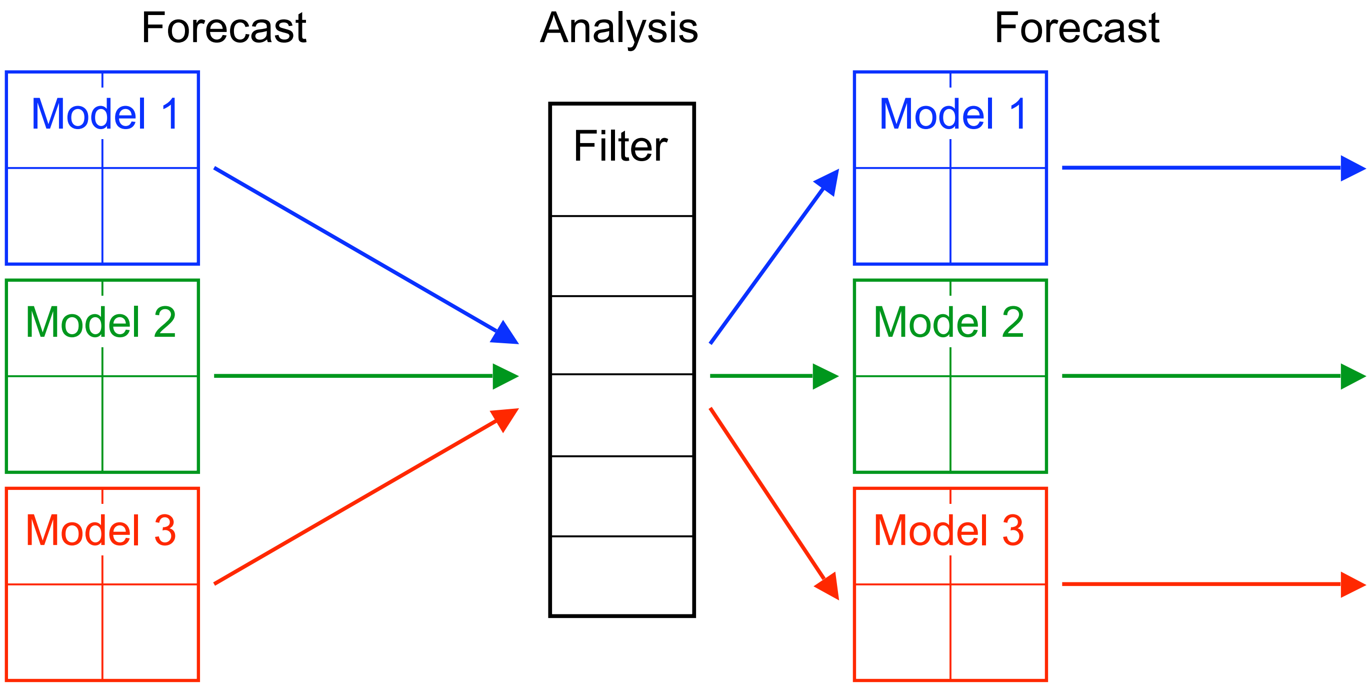
Extension for data assimilation

Also needed:

Observation routines called by PDAF



2-level Parallelism



1. Multiple concurrent model tasks
2. Each model task can be parallelized

- Analysis step is also parallelized

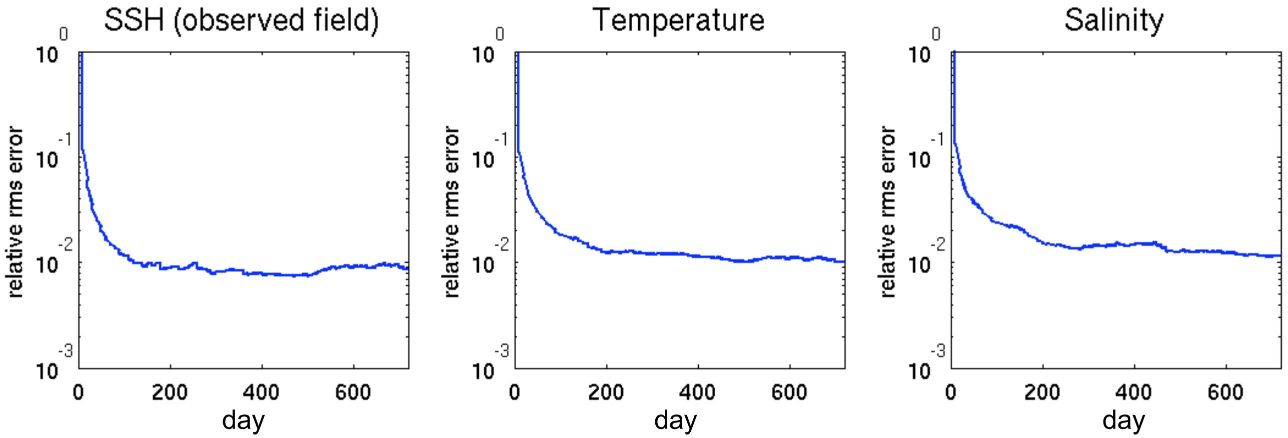
Application Example

Test case: „Twin Experiment“

- FEOM (Finite Element Ocean Model)
- North Atlantic, 1 degree resolution, 20 z-levels
- Assimilate synthetic sea level observations over 2 years
- Data available each 10 days

Assimilation impact

improve model fields by 2 orders of magnitude



Parallel Performance

Use between 64 and 4096 processors of SGI Altix ICE cluster (Intel processors)

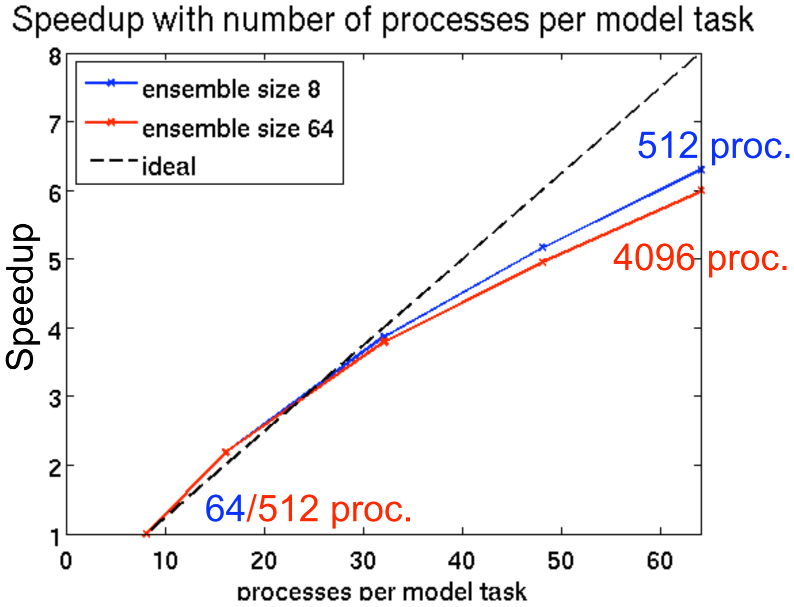
94-99% of computing time in model integrations

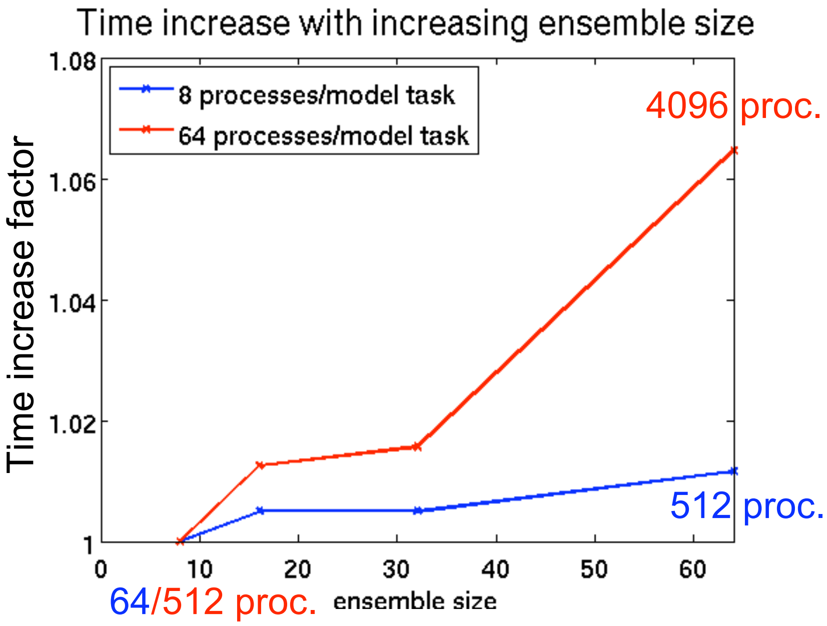
Speedup: Increase number of processes for each model task, fixed ensemble size

- factor 6 for 8x processes/model task
- one reason: time stepping solver needs more iterations

Scalability: Increase ensemble size, fixed number of processes per model task

- increase by ~7% from 512 to 4096 processes (8x ensemble size)
- one reason: more communication on the network





Summary

- Parallel Data Assimilation Framework PDAF

  A tool providing

  - ➢ Simplified implementation of assimilation systems (parallelization, filter algorithms, „fixes“)
  - ➢ Separation of model and assimilation algorithm
  - ➢ Flexibility: Different assimilation algorithms and data configurations within one executable
  - ➢ Full utilization of parallelism in models and filters

Current filter algorithms in PDAF

- Ensemble Kalman filter (EnKF, Evensen, 1994)
- SEIK filter (Pham et al., 1998a)
- SEEK filter (Pham et al., 1998b)
- ETKF (Bishop et al., 2001)
- LSEIK filter (Nerger et al., 2006)
- LETKF (Hunt et al., 2007)

Thank you!

PDAF is open source:

available upon request (not yet downloadable ☹)

More information at

www.awi.de/en/go/pdaf

Requirements

- Fortran compiler (gfortran works!)
- MPI (OpenMPI works!)
- BLAS & LAPACK
- make

- No Matlab version!