Sequential data assimilation on high-performance computers with the Parallel Data Assimilation Framework PDAF

Lars Nerger

Alfred Wegener Institute for Polar and Marine Research
Bremerhaven, Germany

and

Bremen Supercomputing Competence Center BremHLR

lars.nerger@awi.de

BremHLR
Kompetenzzentrum für Höchstleistungsrechnen Bremen

AWI

Overview

- Sequential data assimilation
  - ➢ Ensemble-based Kalman filters
- Parallel Data Assimilation Framework PDAF
- Parallel performance of PDAF
- Application examples

Sequential Data Assimilation

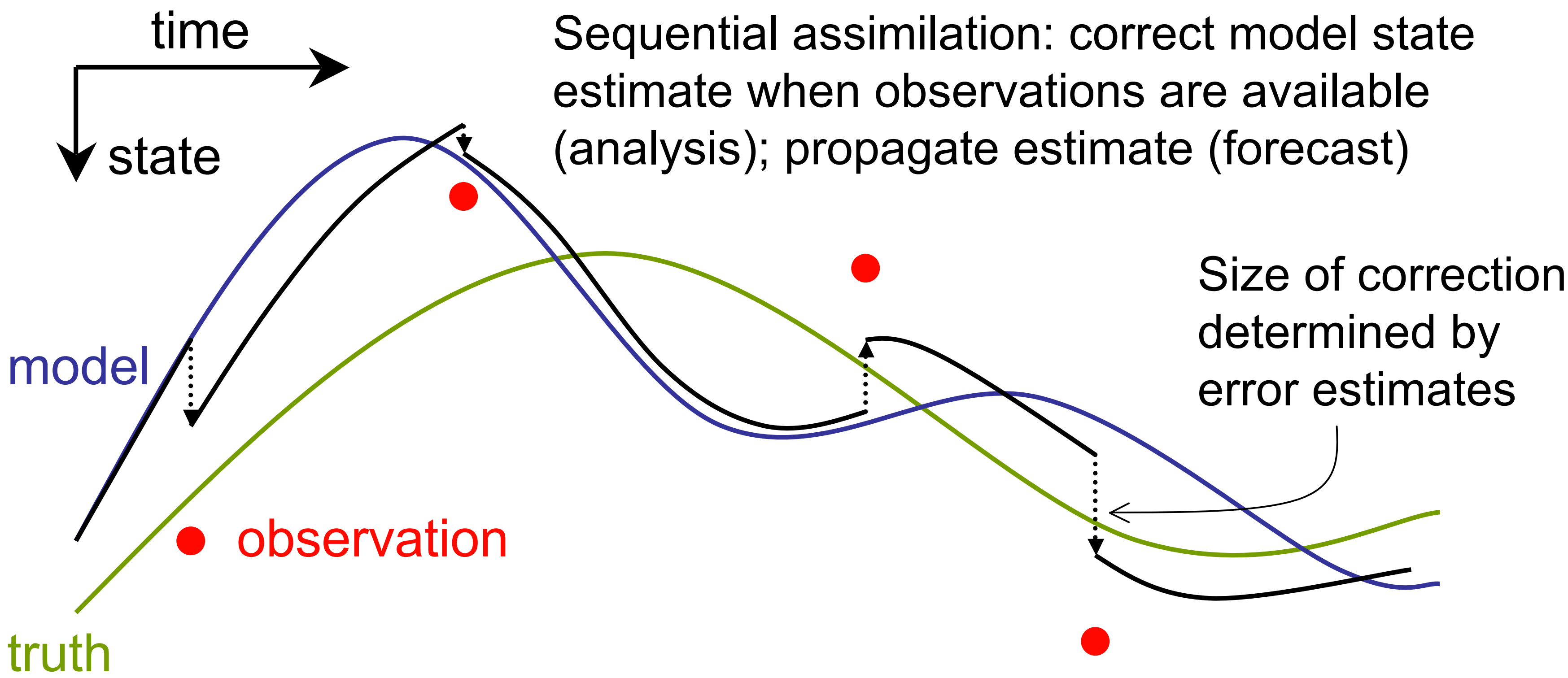
Data Assimilation

- Optimal estimation of system state:
  - initial conditions (for weather forecasts, …)
  - trajectory (temperature, concentrations, …)
  - parameters (growth of phytoplankton, …)
  - fluxes (heat, primary production, …)
  - boundary conditions and ‘forcing’
- Characteristics of system:
  - high-dimensional numerical model - $\mathcal{O}(10^7)$
  - sparse observations
  - non-linear

Sequential Data Assimilation

Consider some physical system (ocean, atmosphere,…)

Sequential assimilation: correct model state estimate when observations are available (analysis); propagate estimate (forecast)

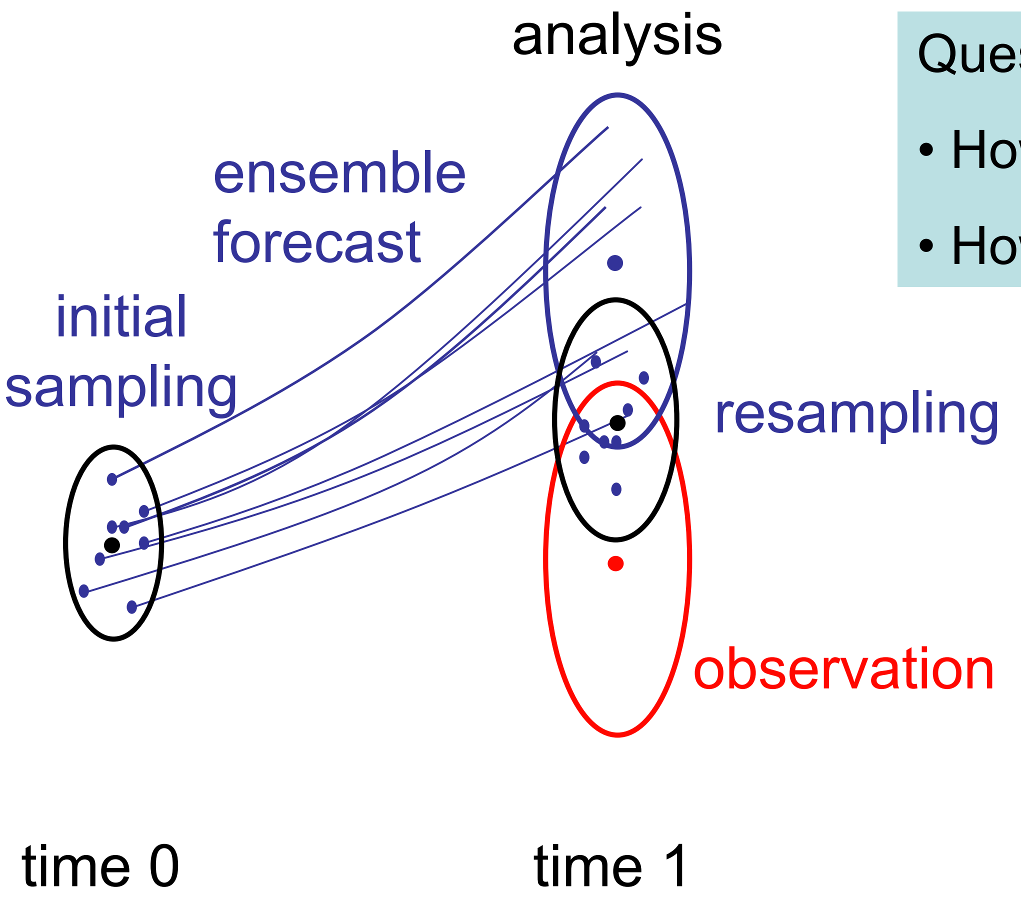


Kalman Filters (Kalman, 1960)

- Optimal estimation problem
- Assume errors to be Gaussian distributed
  - ➢ Analysis is combination of two Gaussian distributions
  - ➢ Analysis is variance-minimizing
- Express problem in terms of mean state **x** and state error covariance matrix **P**
- Propagate matrix **P** by (linearized) model
- Issues:
  - Nonlinearity will not conserve Gaussianity
  - Storage of state covariance matrix can be unfeasible
  - Evolution of covariance matrix extremely costly
  - Reduce cost: simplify dynamics and/or approximate **P**

Ensemble-based Kalman Filter

Approximate probability distributions by ensembles



Questions:

- How to generate initial ensemble?
- How to resample after analysis?

Some filters:

- EnKF
  (Evensen1994)
- SEIK
  (Pham et al. 1998)
- ETKF, EAKF, …
  (2001 - …)

Computational and Practical Issues

- Huge amount of memory required
  (model fields and ensemble matrix)
- Huge requirement of computing time
  (ensemble integrations)
- Natural parallelism of ensemble integration exists
  - but needs to be implemented
- Existing models often not prepared for data
  assimilation

Parallel Data Assimilation Framework

Motivation

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

- Goals
  - 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“ for filters, which showed good performance in studies

PDAF: Considerations for Implementation

Logical separation of problem

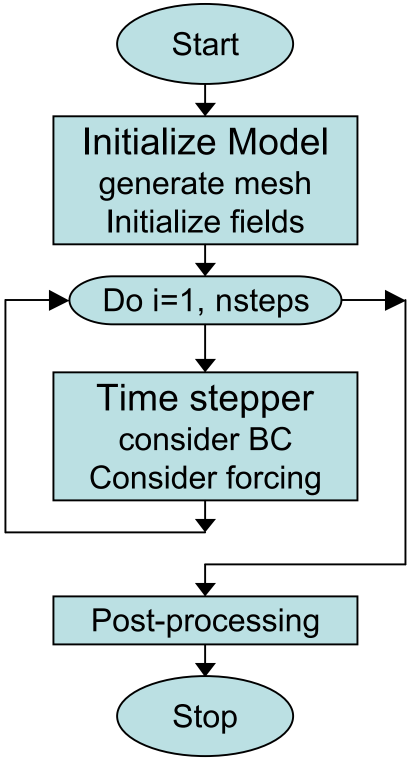
Model		Filter		Observations
initialization time integration post processing	state vector ⟷ time	Initialization analysis re-initialization	state vector ⟷ obs. vector	obs. vector obs. operator obs. error

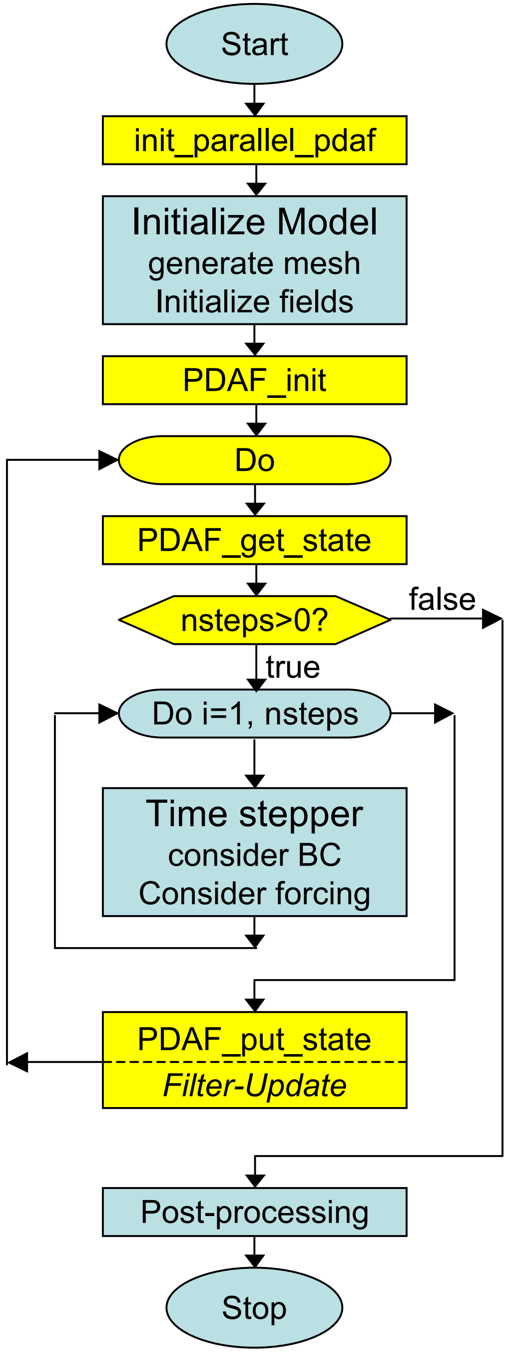
Core of PDAF

Further considerations

- Combination of filter with model with minimal changes to model code
- Control of assimilation program coming from model
- Simple switching between different filters and data sets
- Complete parallelism in model, filter, and framework

Model





Extension for data assimilation

PDAF also has an offline-mode:

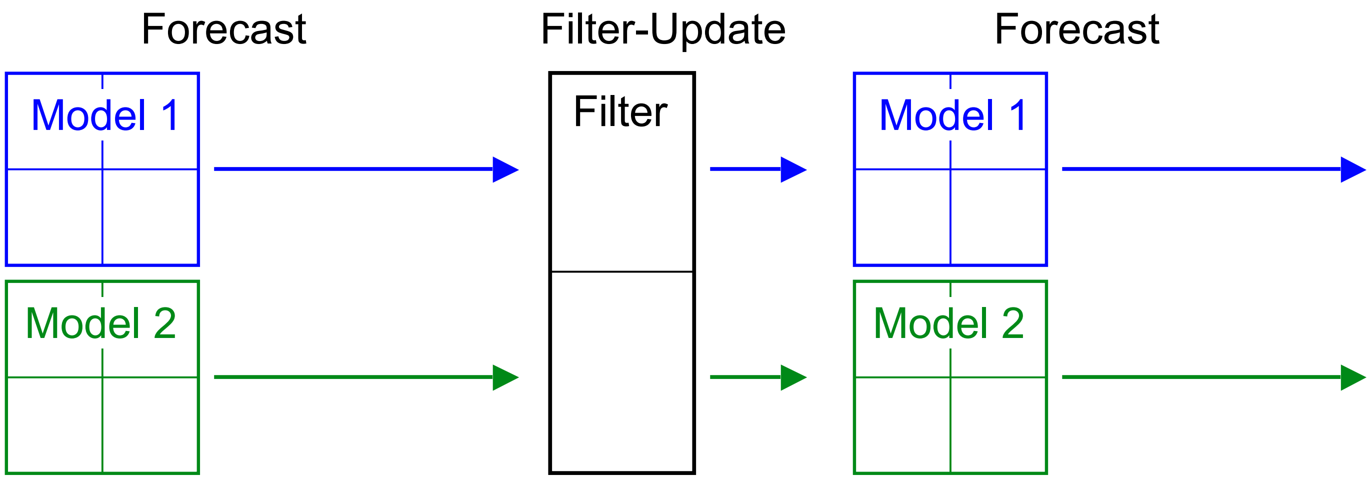
A. Run forecasts with model

B. Read model outputs, perform analysis & write restart files

PDAF interface structure

- Interface independent of filter
  (except for names of user-supplied subroutines)
- User-supplied routines for elementary operations:
  - field transformations between model and filter
  - observation-related operations
  - filter pre/post-step
- User supplied routines can be implemented
  as routines of the model
  (e.g. share common blocks or modules)

2-level Parallelism



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

- Filter-update is parallel
- 2 parallelization strategies:
  distribute ensemble members or state in sub-domains

Current KF algorithms in PDAF

- Ensemble Kalman filter (EnKF, Evensen, 1994)
  - original ensemble-based KF
  - simplest formulation of ensemble-based KFs
- SEIK filter (Pham et al., 1998)
  - very efficient ensemble-based KF
- LSEIK filter (Nerger et al., 2006)
  - localized analyses for better filter performance
- SEEK filter (Pham et al., 1998)
  - explicit low-rank (error-subspace) formulation
  - linearized error forecast

Parallel Performance of PDAF

Parallel performance of PDAF

- Performance tests on

  SGI Altix ICE at HRLN (German "High performance computer north")

  nodes: 2 quad-core Intel Xeon Harpertown at 3.0GHz
  network: 4x DDR Infiniband
  compiler: Intel 10.1, MPI: MVAPICH2

- Ensemble forecasts

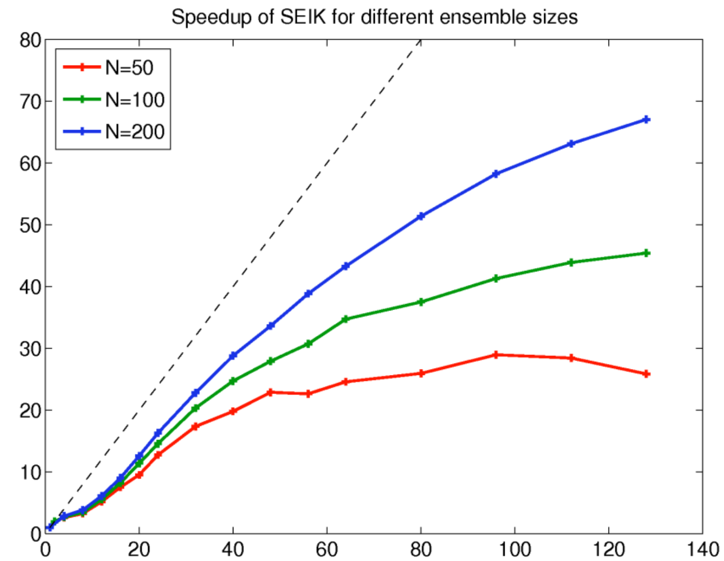
  - are naturally parallel

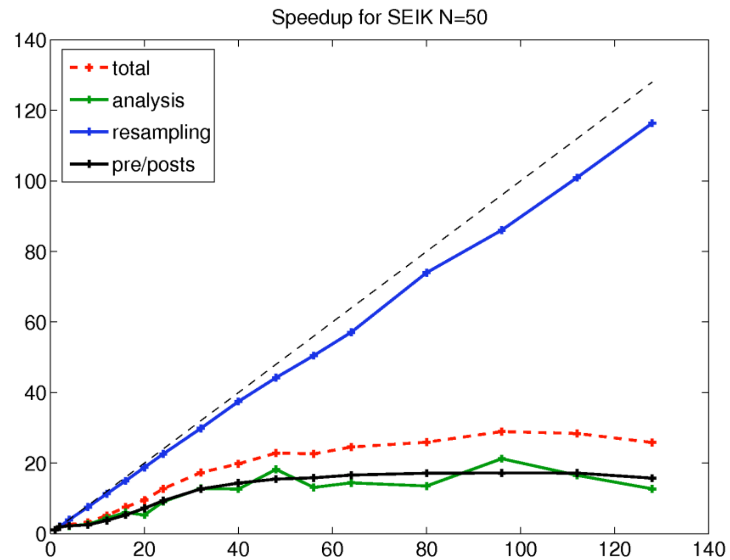
  - dominate computing time
    E.g. parallel forecast over 10 days: 45s
    SEIK with 16 ensemble members: 0.1s
    LSEIK with 16 ensemble members: 0.7s

  - parallel efficiency near 1

Speedup of SEIK with domain decomposition

- Test only assimilation without model dynamics
- SEIK performs global optimization
  - better speedup for larger ensembles
  - resampling is local, but no ideal speedup (MKL library?)
  - analysis and pre/poststep show very small speedup
  - ➢ behavior seems to be due to network latency of the machine used

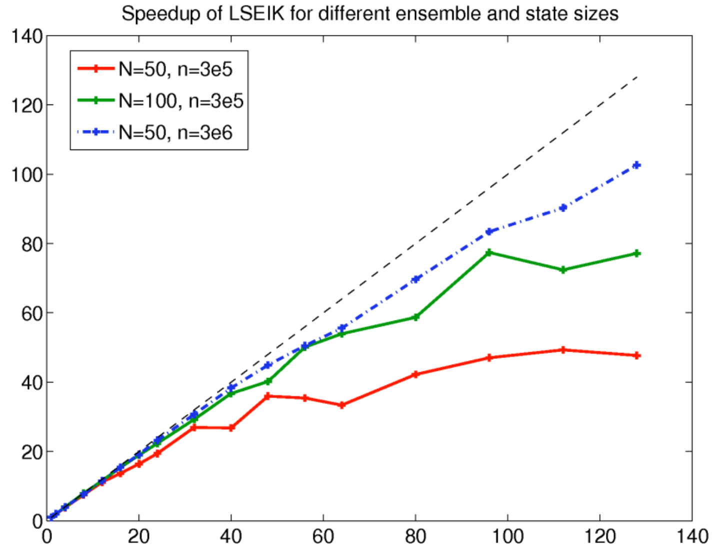


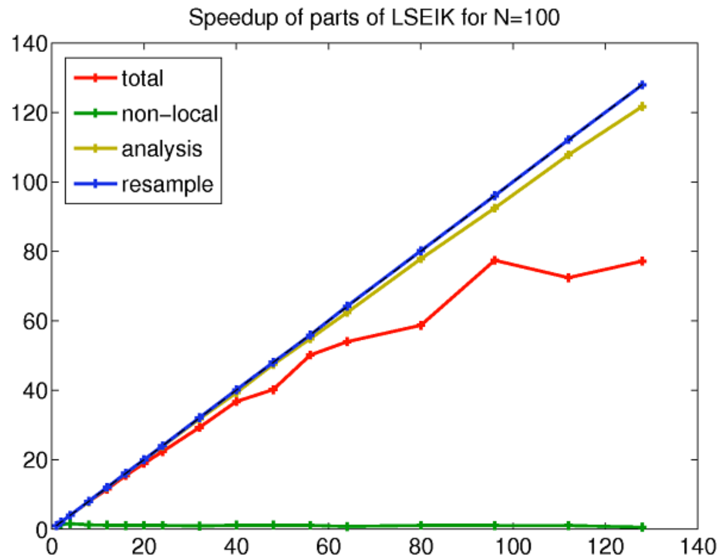


State dimension n = 3,000,000
Observations m = 30,000
Ensemble size N

Speedup of LSEIK with domain decomposition

- LSEIK performs sequence of local optimizations on sub-subdomains defined by influence radius for observations
  - near-ideal speedup for analysis step and resampling (ensemble transformation)
  - total speedup is limited by
    - ➢ non-local gathering of observation-state residuals
    - ➢ pre/poststep





State dimension n = 300,000
Observations m = 30,000
Ensemble size N

Application examples

- Assimilation of satellite altimetry
  (Project Tandem, @ AWI T. Janjic Pfander)
  - ➢ with finite element ocean model FEOM
  - ➢ utilize information from tandem mission of Topex/Poseidon and Jason 1
- Ocean chlorophyll assimilation into global NASA Ocean Biogeochemical Model (with Watson Gregg, NASA GSFC)
  - ➢ Generation of daily re-analysis maps of chlorophyll at ocean surface
- Coastal assimilation of ocean surface temperature
  (within project “DeMarine Environment”, AWI and BSH)
  - ➢ Improve operational forecast skill, e.g. for storm surges

PDAF is available!

- With a restricted GPL-license
- Upon request (not yet downloadable ☹)
  - ➢ Mail me (Lars.Nerger@awi.de)
  - ➢ Go to

    www.awi.de/en/go/pdaf

    to get contact information

- Distributed is the source code of PDAF together with an example implementation

Requirements

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

- I don‘t have a Matlab version!

Summary

- Sequential data assimilation is not serial

⇨ Mixed parallel efficiency of ensemble-based Kalman filtering (forecasts & analysis/resampling)

- Parallel Data Assimilation Framework PDAF
  - Simplified implementation of assimilation systems
  - Flexibility: Different assimilation algorithms and data configurations within one executable
  - Full utilization of parallelism in models and filters
  - Available upon request

Thank you!