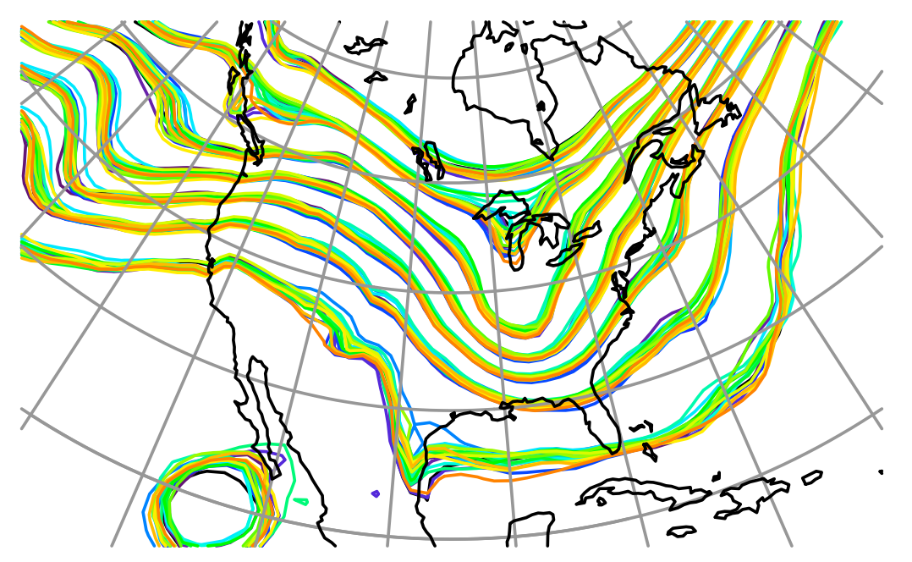


Parallel Implementation of Ensemble Filter Algorithms for Data Assimilation

N.Collins, J.Anderson, T.Hoar, K.Raeder, H.Liu

National Center for Atmospheric Research,
Institute for Mathematics Applied to Geosciences
`nancy@ucar.edu`

Introduction

We describe a parallelized implementation of an Ensemble Filter (EF) algorithm for Data Assimilation (DA) which scales well on current cluster sizes.

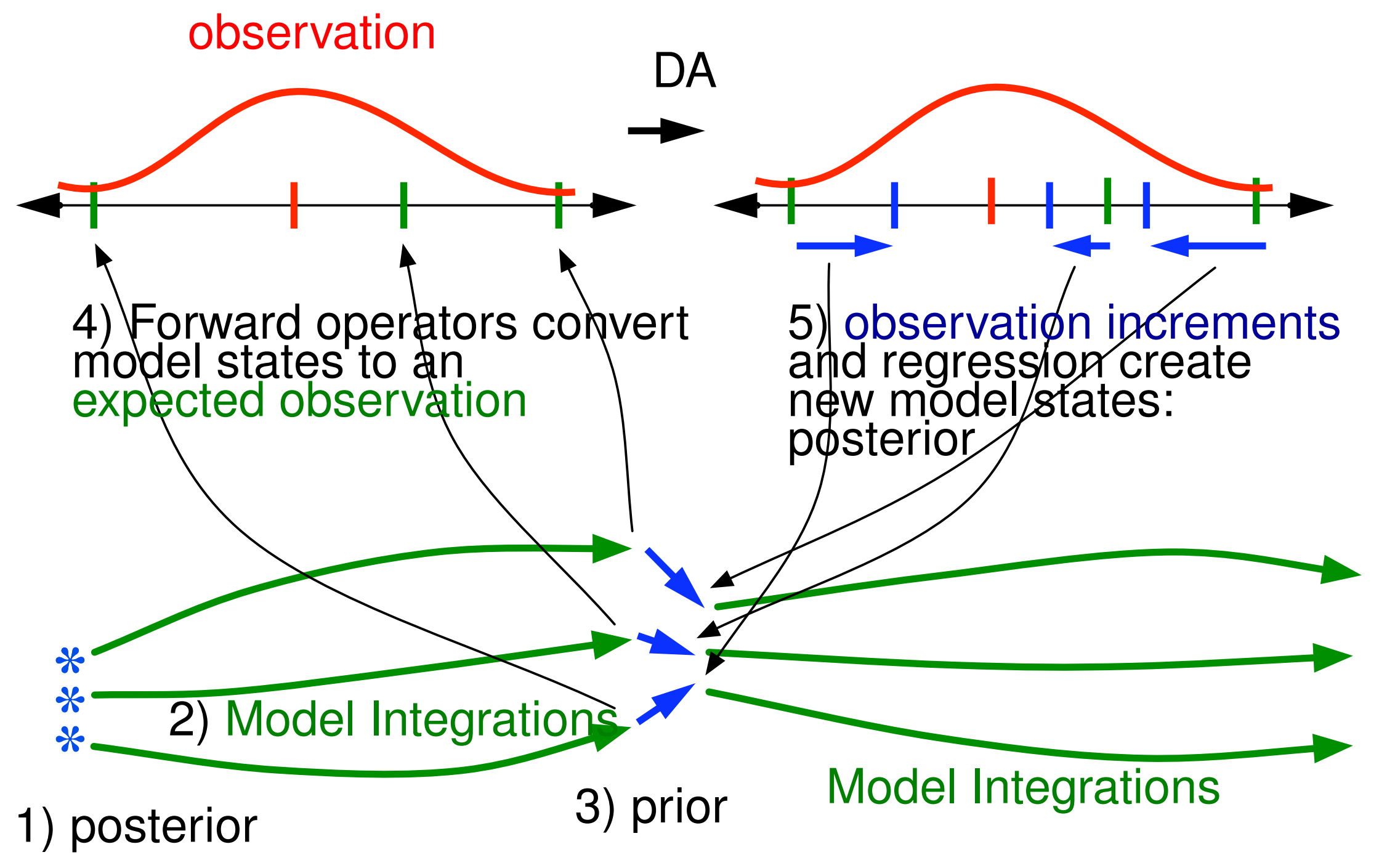
Running ensembles of models is generally an embarrassingly-parallel problem; each ensemble member runs with a different input data file or set of parameters and does not interact with other members. However, the data assimilation step in the EF algorithm requires collecting corresponding data values from all ensemble members, computing the mean and variance, and perturbing them to better match observed values. This requires slicing the corresponding data from each ensemble member along an orthogonal axis.

The underlying models which are being combined with data are frequently chaotic, meaning small differences are magnified by the system, so parallelizing the algorithms should not in any way introduce numerical differences in the results.

We describe important considerations in the development of the parallel algorithm, show scaling numbers, talk about the DART testbed as a general purpose DA tool, and discuss envisioned extensions into the coming petascale world.

1. What is Data Assimilation?

The process of Data Assimilation (DA) combines observations of a system with predictions from a numerical forecast model of that system. DA is frequently used in Numerical Weather Prediction (NWP) systems to construct a set of initial conditions before running forecasts further into the future. DA can also be used for many other purposes including evaluating errors in the numerical model, finding appropriate values for model parameters, and designing better observational systems.


observation
DA
4) Forward operators convert model states to an expected observation
5) observation increments and regression create new model states: posterior
2) Model Integrations
Model Integrations
1) posterior
3) prior

2. DART

The Data Assimilation Research Testbed (DART) is a community software facility that allows users to easily try EF techniques. The distributed code includes many models with various levels of complexity, various sets of observations, and skeleton code to guide users in adding their own models or new observation types. The DART algorithms are designed so that incorporating new models and new observation types requires minimal coding of a small set of interface routines, and does not require modification of the existing model code. The expected scaling of the DART parallel algorithm is independent of the forecast model.

3. Transposes

Each PE can store the entire state vector for a subset of the ensemble members, or all ensemble members for a subset of the state vector. To go between these two representations requires an All-To-All communication.

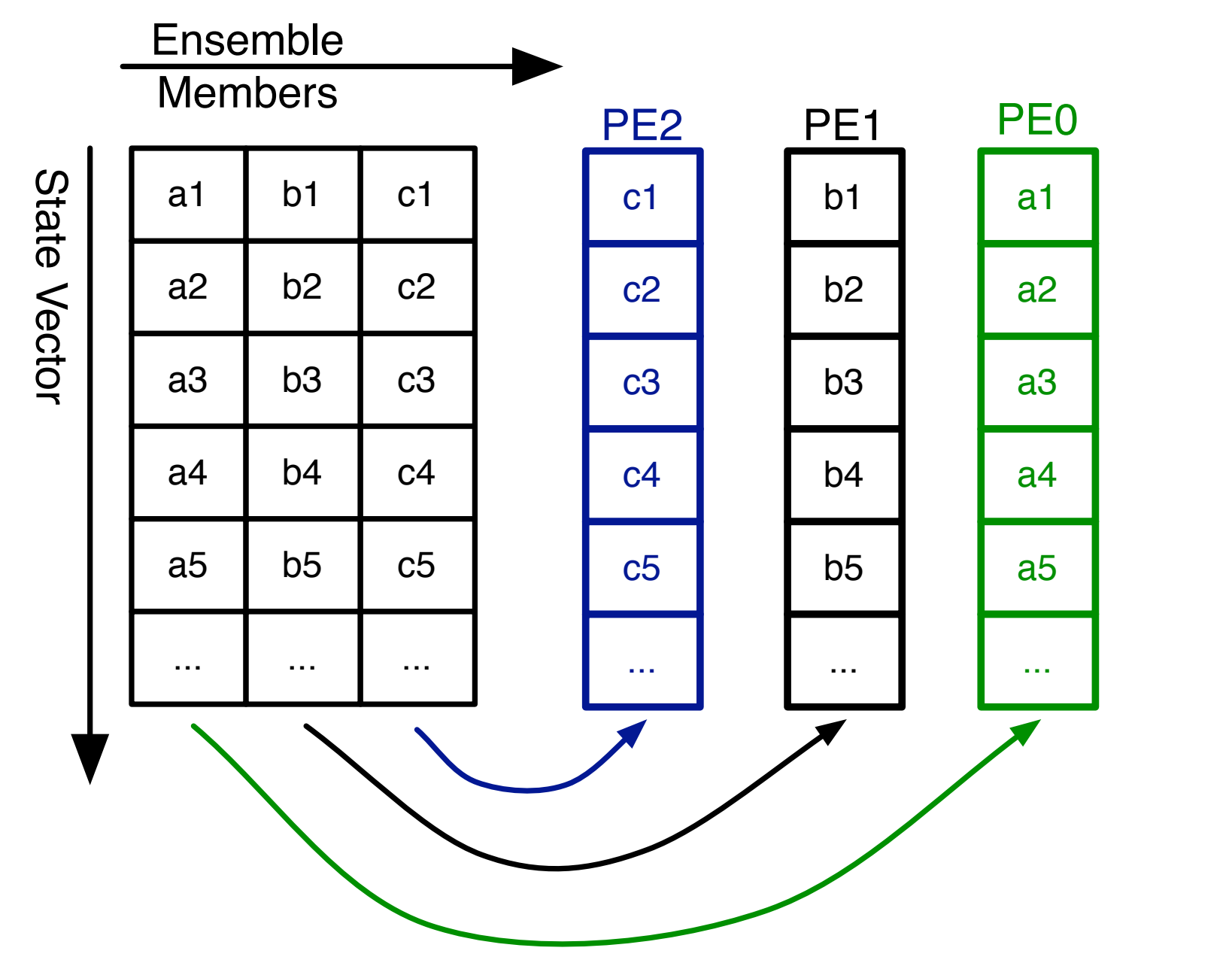


Figure 1: *Each task (PE) has the entire state vector for a subset of the ensemble members.*

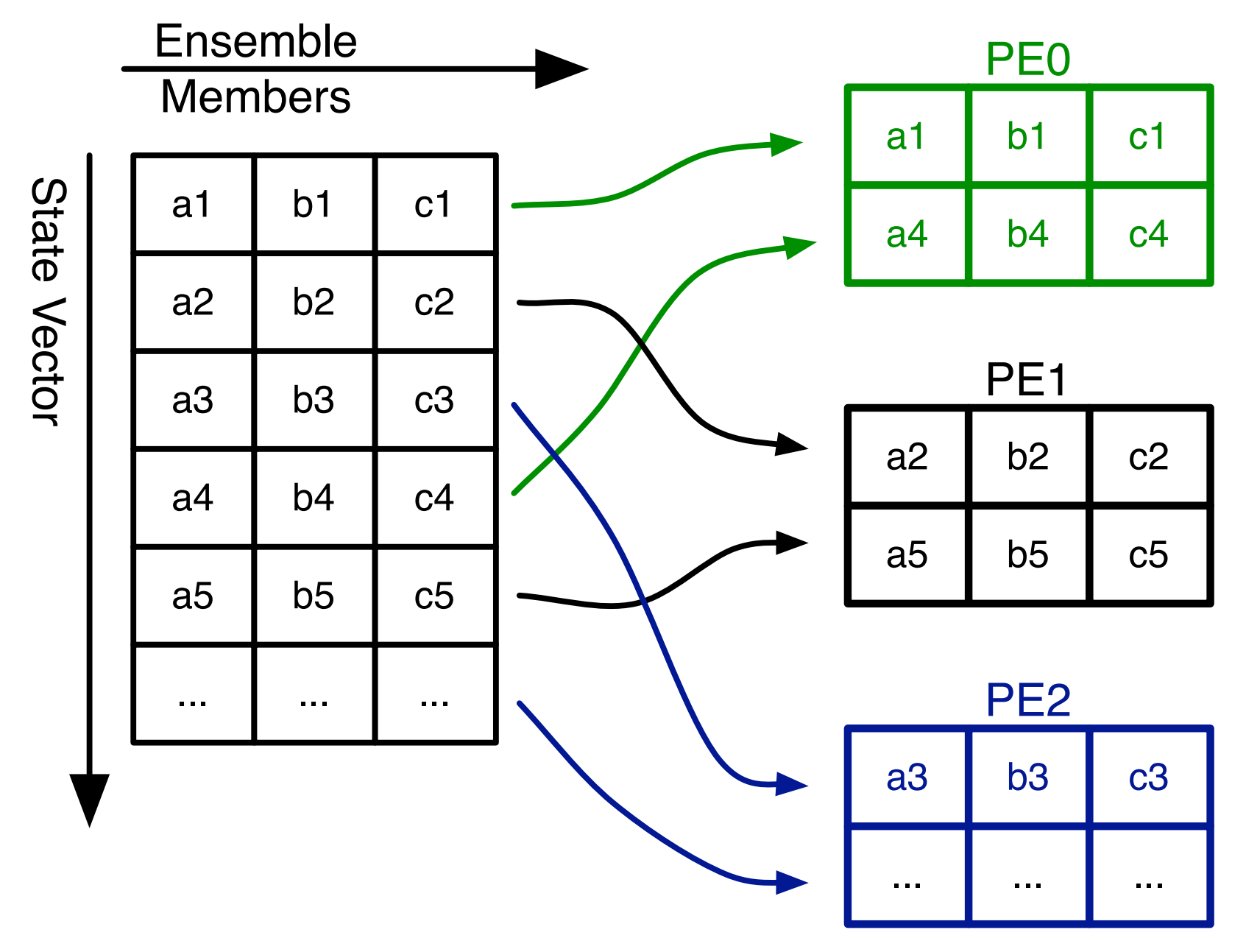


Figure 2: *Each task (PE) has a subset of the state vector for all ensemble members.*

4. Serial vs. Parallel

Observations are assimilated in a sequential order to maintain bit-wise reproducibility.

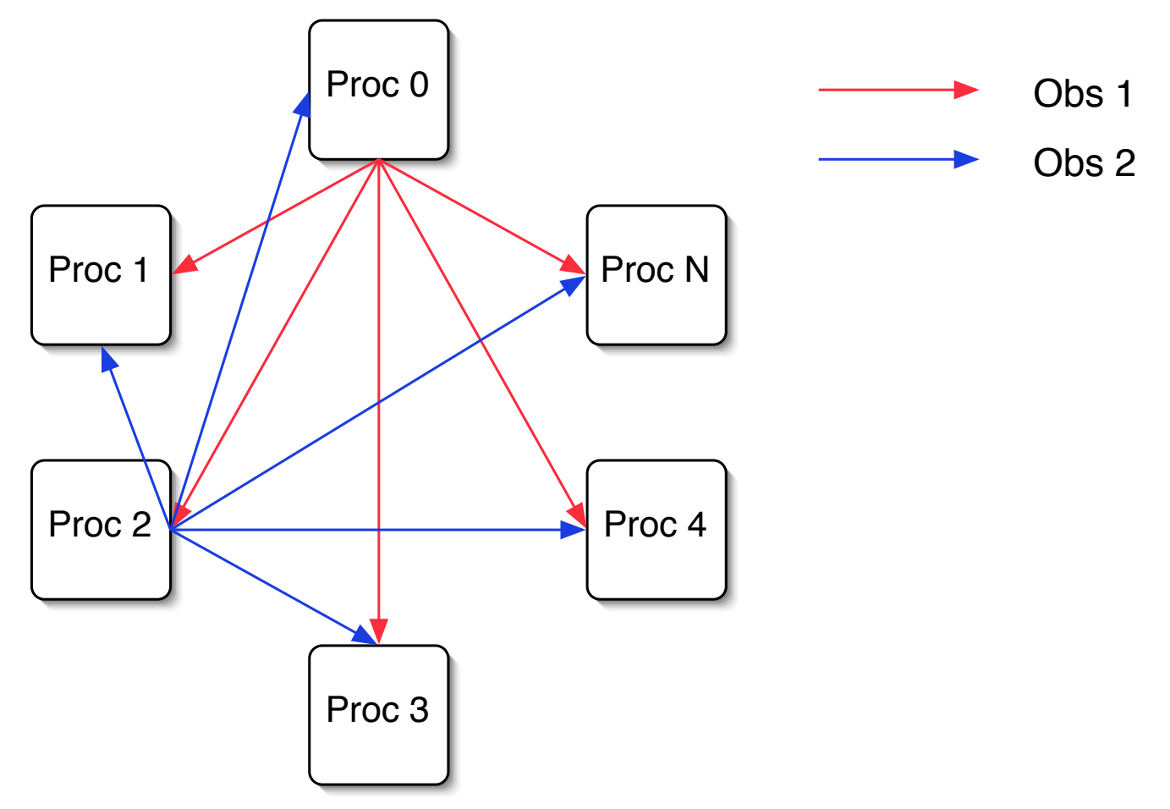


Figure 3: *The tasks take turns broadcasting the increments for each observation, to maintain bitwise reproducibility of the results for any number of tasks.*

The observations are distributed across all tasks. The task with the next observation computes the update increments, broadcasts them to the other tasks, and then in parallel they compute the impact of that observation on the state vector. With current processor counts this is not the rate limiting step, but looking ahead to the envisioned petascale computing environments this step of the algorithm will need to be enhanced.

The overall execution flow has several options for parallelism. The two most common cases are illustrated below, where the assimilation program `filter` is a parallel MPI program, and the model is either a parallel MPI program or a single-task program and is unaware that multiple copies are being run simultaneously.

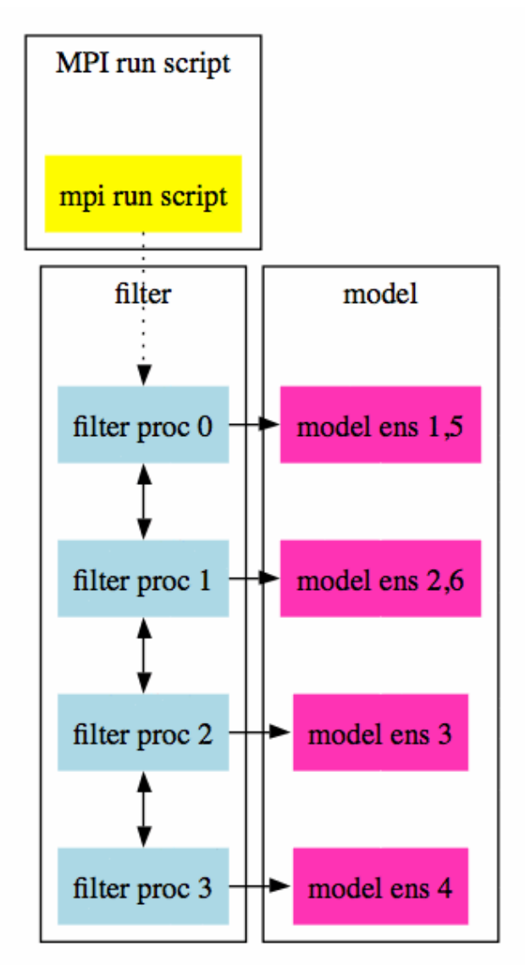


Figure 4: *The* `filter` *program does the assimilation of the observations, and runs multiple copies of the model.*

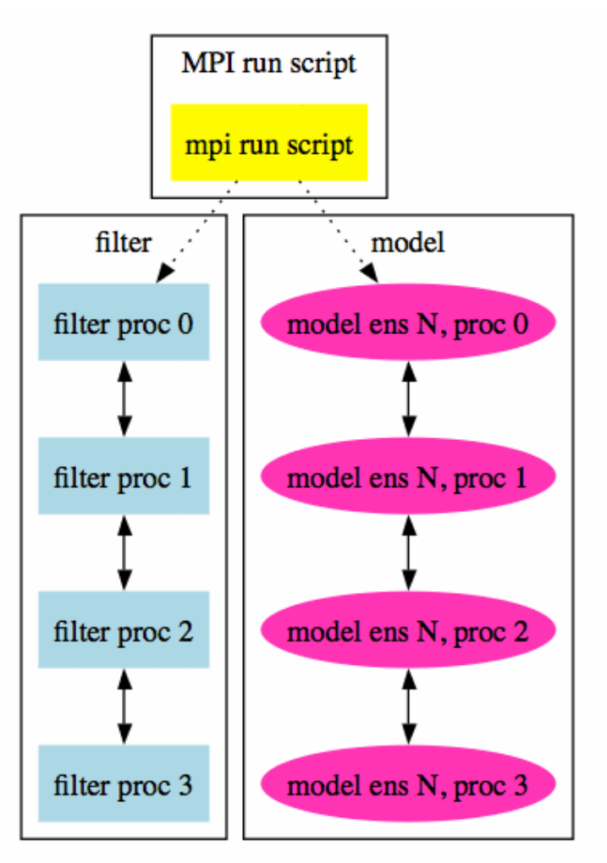


Figure 5: *The* `filter` *program does the assimilation of the observations, and the MPI job script runs the MPI-parallel model multiple times in succession.*

5. Timing results

Scaling runs were done using a state-of-the-art global atmospheric climate model (CAM) at low and medium resolutions on a commodity Intel-based Linux cluster from Aspen Systems, an Intel-based Linux cluster from IBM, and a Power 5+ AIX system with a high-speed switch. The timing results shown below are from the Aspen Systems cluster, and are representative of the results obtained on the other systems.

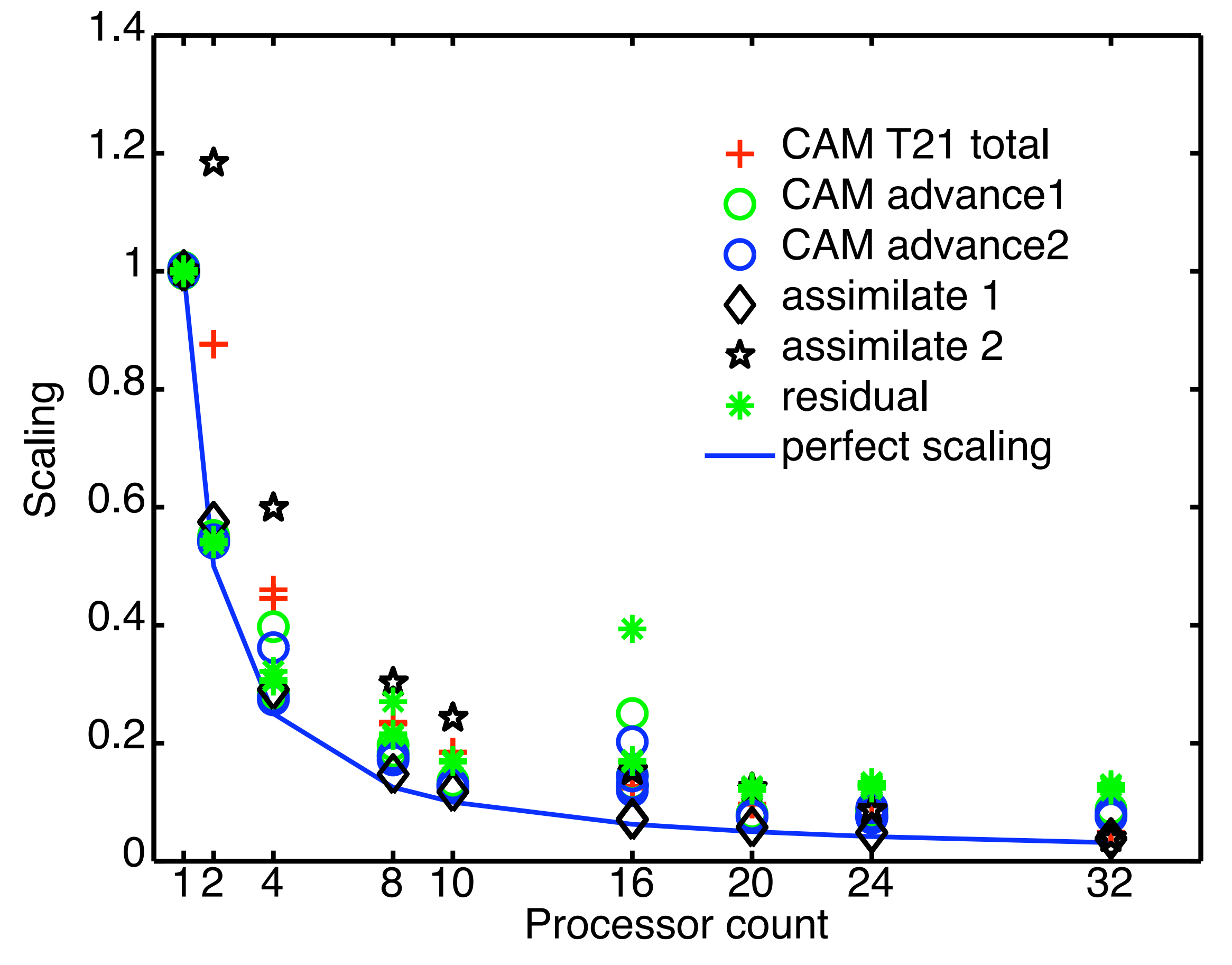


Figure 6: *Scaling on a 16-node Aspen Systems Linux cluster. Each node is a dual-processor, 3.2Ghz, IA-32 EM64T with 4GB shared memory. The times are for a 20 member CAM T21 data assimilation (state vector length $\approx$ 320,000) with two 6-hour model advances, assimilating about 210,000 observations.*

6. For further information

Our DART web site is: http://www.image.ucar.edu/DAReS/DART There you will find information about how to download the latest revision of DART from our subversion server, information on a full DART tutorial (included with the distribution), and contact information for the DART development group.

References

[1] Anderson, J., Collins, N., *Scalable Implementations of Ensemble Filter Algorithms for Data Assimilation*, to appear in Journal of Atmospheric and Oceanic Technology.