

Parallel Implementations of Ensemble Data Assimilation for Atmospheric Prediction

Jeffrey Anderson, Helen Kershaw, and Nancy Collins

*Institute for Mathematics Applied to Geophysics,
National Center for Atmospheric Research
Email: jla@ucar.edu*

Ensemble Atmospheric Prediction

Prediction Model

O(100 million) gridded variables.

Highly optimized.

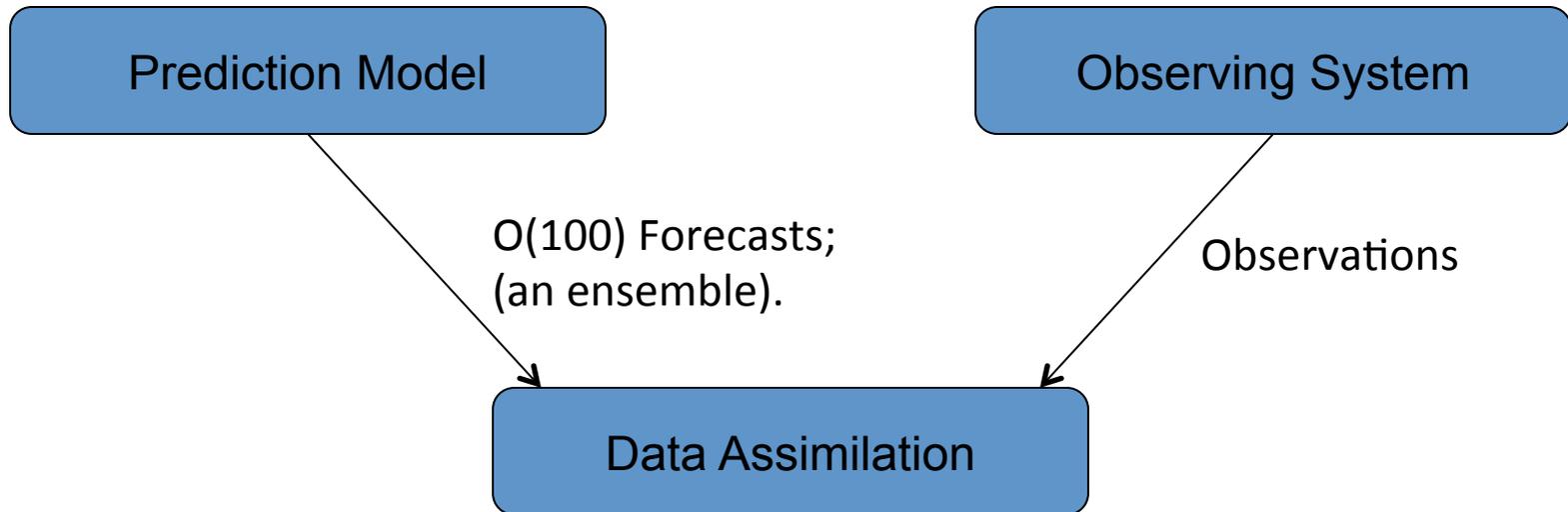
Ensemble Atmospheric Prediction

Prediction Model

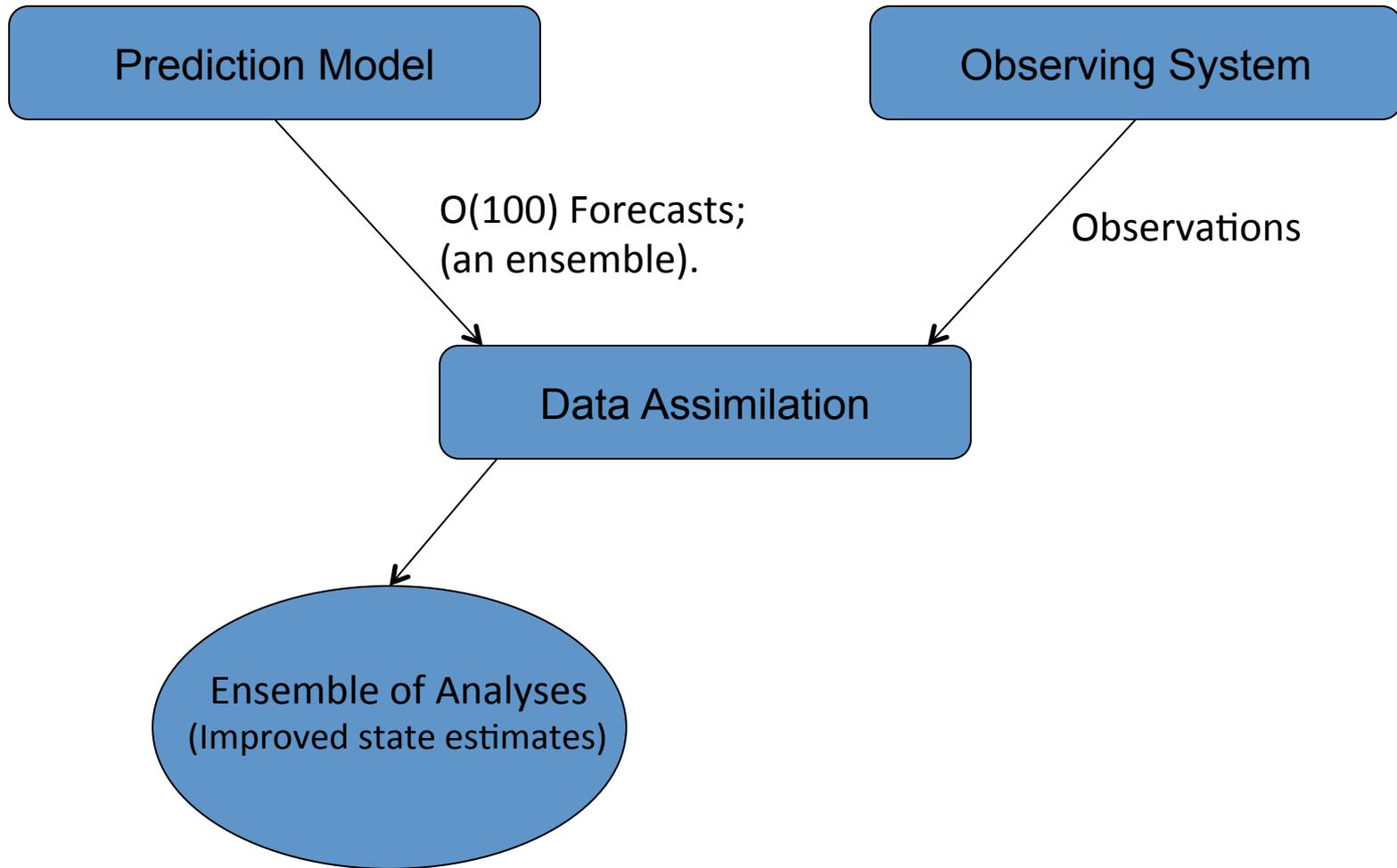
Observing System

O(100 million)
observations per day.

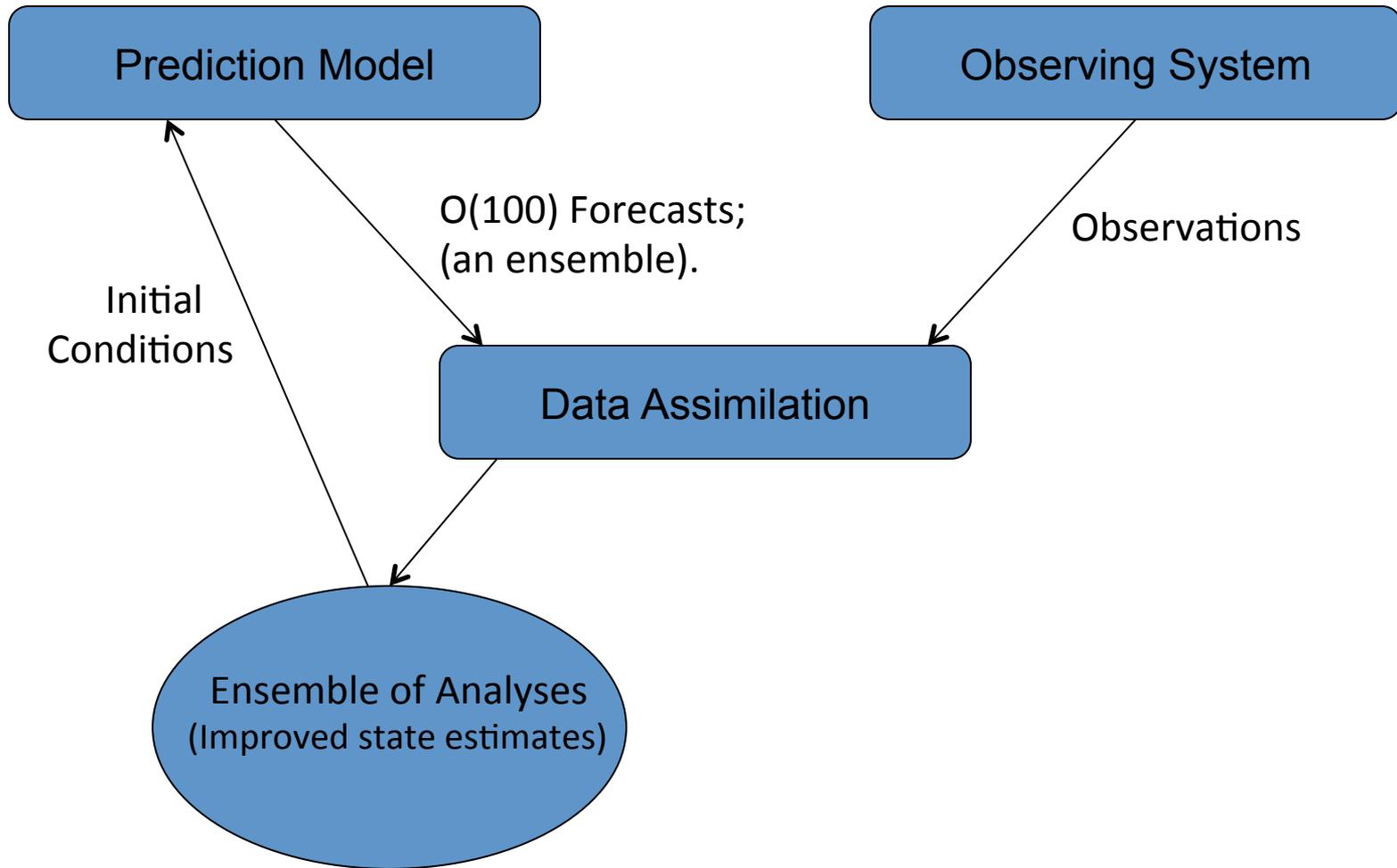
Ensemble Atmospheric Prediction



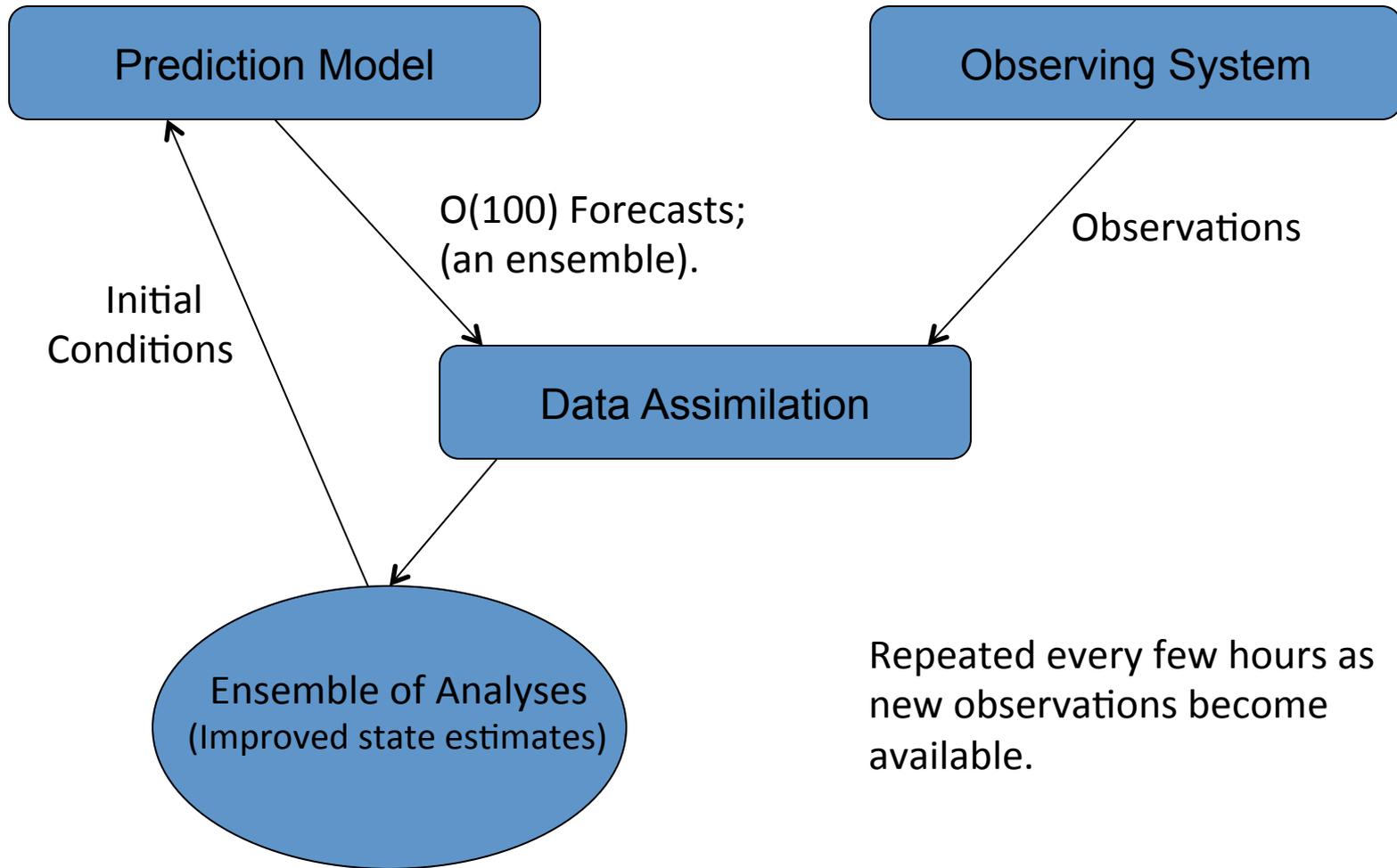
Ensemble Atmospheric Prediction



Ensemble Atmospheric Prediction

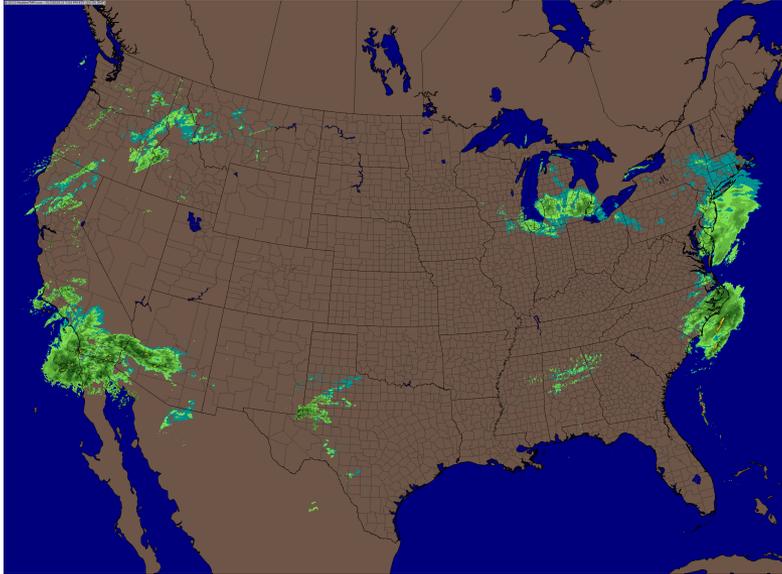


Ensemble Atmospheric Prediction

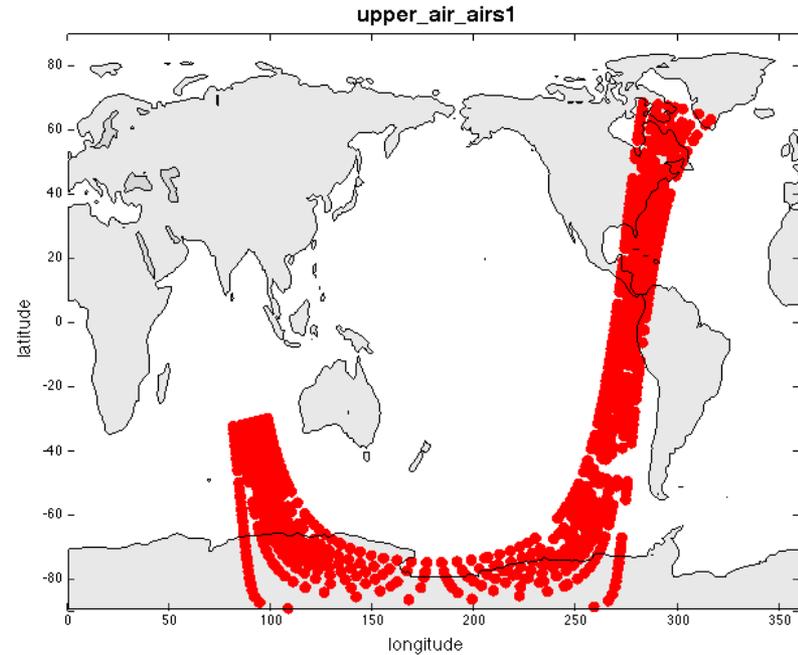


Repeated every few hours as new observations become available.

Observations Irregular in Time & Space



Radar: Only where precipitating.



Satellite: Controlled by orbit.

The Data Assimilation Research Testbed (DART)

- General purpose ensemble DA tools.
- Must work with any model.
- Must work with any observations.
- Must run efficiently on $O(10,000)$ cores.

Sequential Ensemble Filter

1. Use model to advance **ensemble** (3 members for illustration) to time at which next observations becomes available.

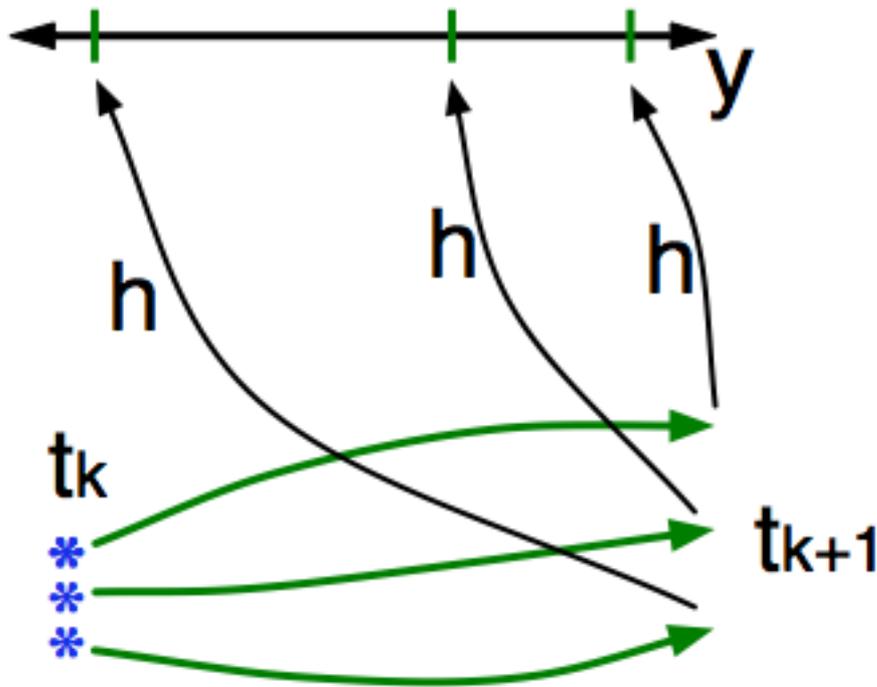
Ensemble state estimate after using previous observations (**analysis**)

Ensemble state at time of next observations (**prior**)



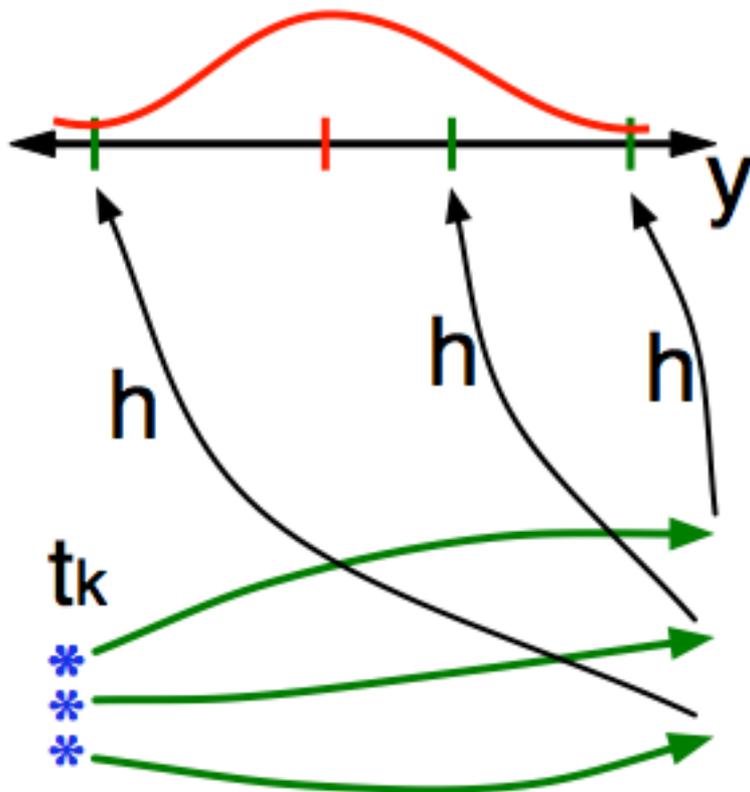
Sequential Ensemble Filter

2. Get prior ensemble sample of independent subset of observations by applying forward operator h to each ensemble member for each observation.



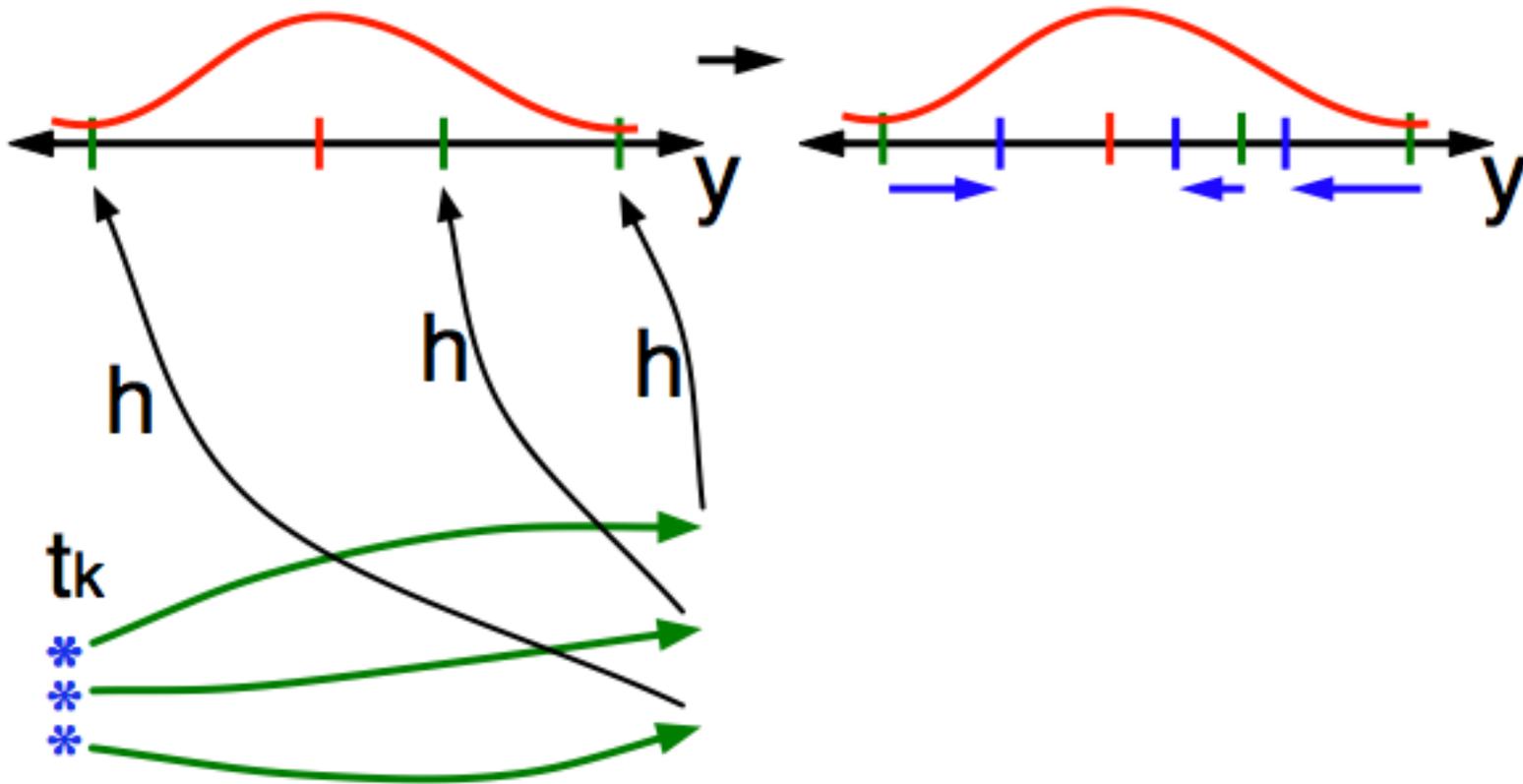
Sequential Ensemble Filter

3. Get **observed values** and **observational error distributions** from observing system.



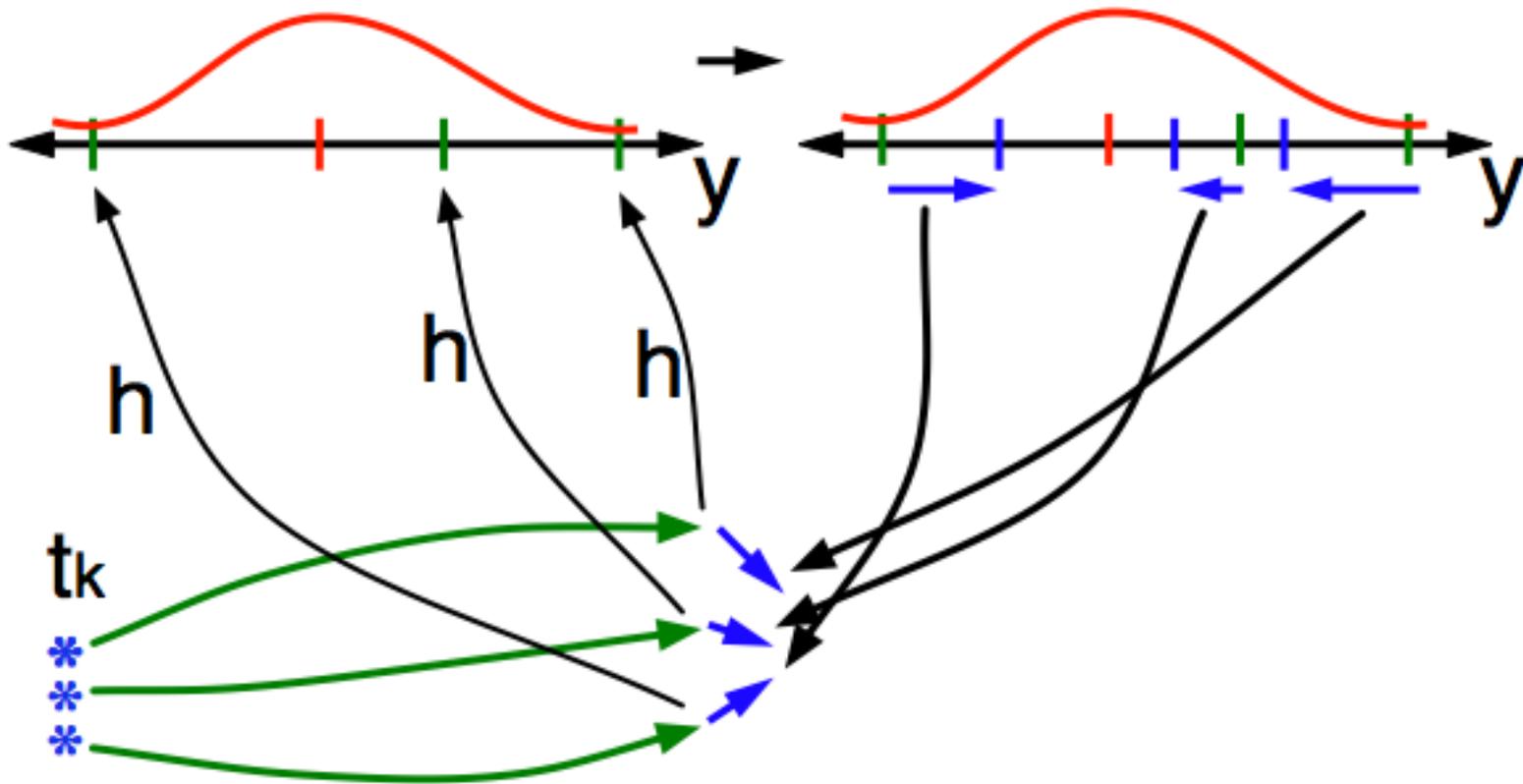
Sequential Ensemble Filter

4. Find the **increments** for the prior observation ensemble.



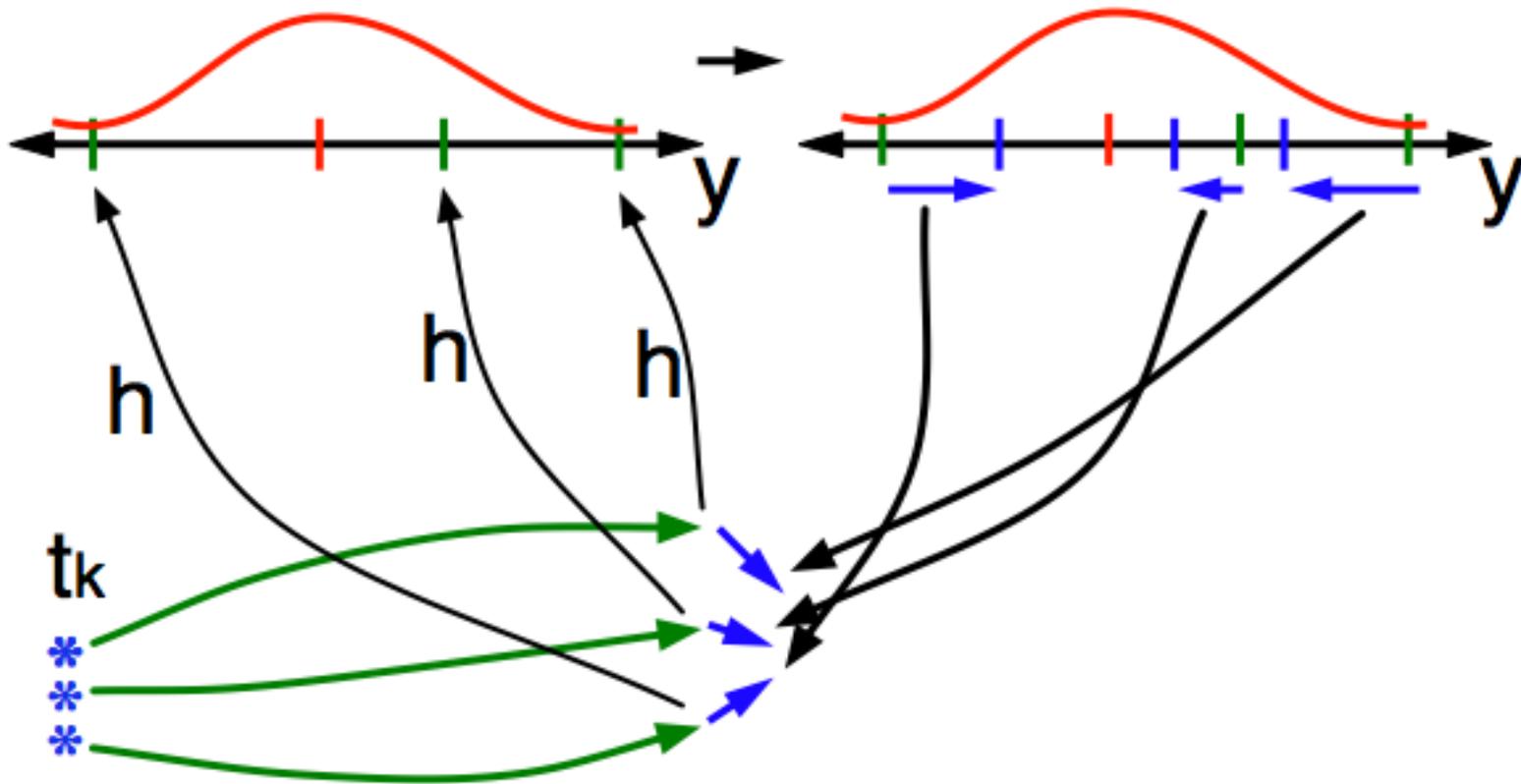
Sequential Ensemble Filter

5. Use ensemble samples to linearly regress observation increments onto state variable increments.



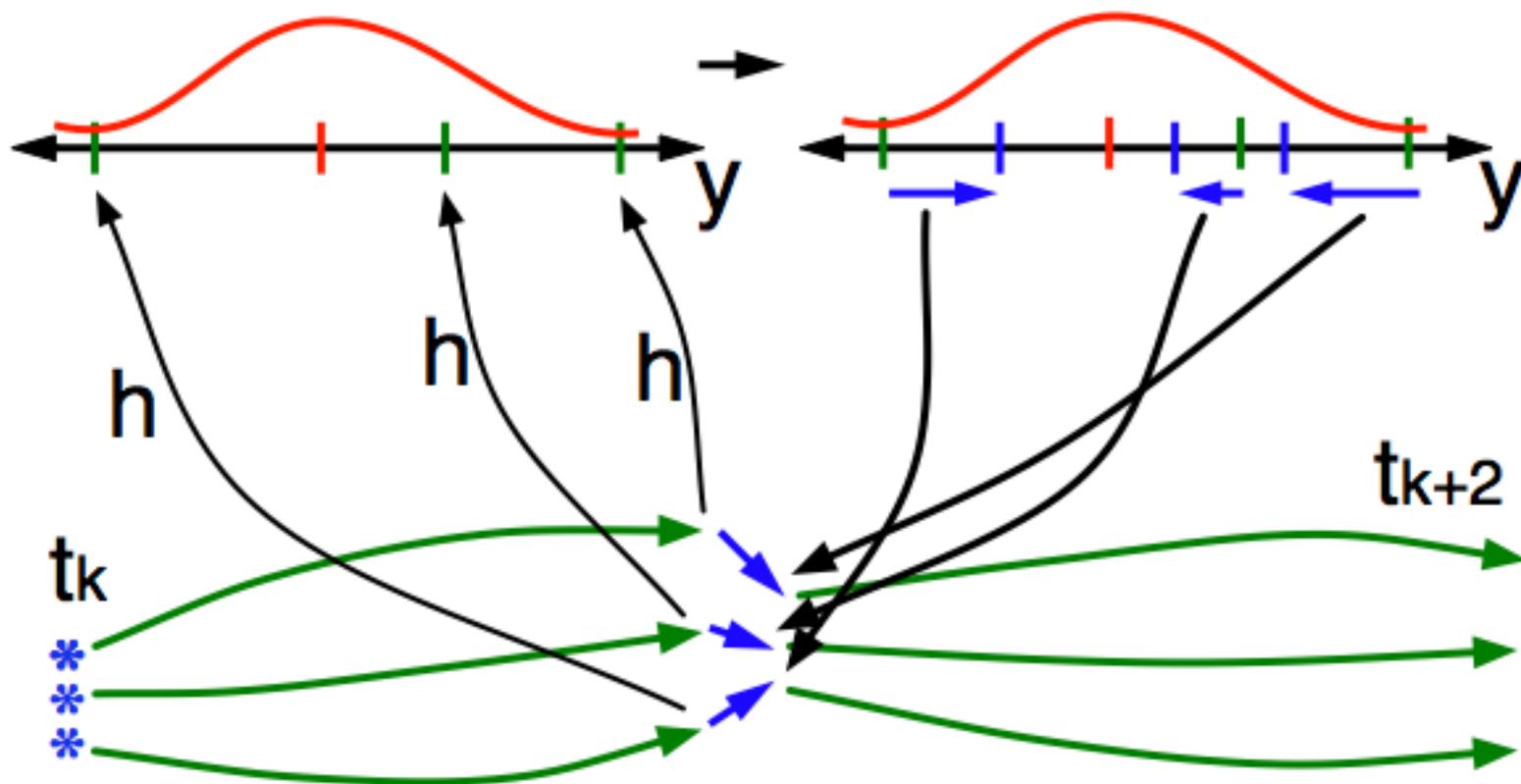
Sequential Ensemble Filter

6. Repeat this for each subset of independent observations.



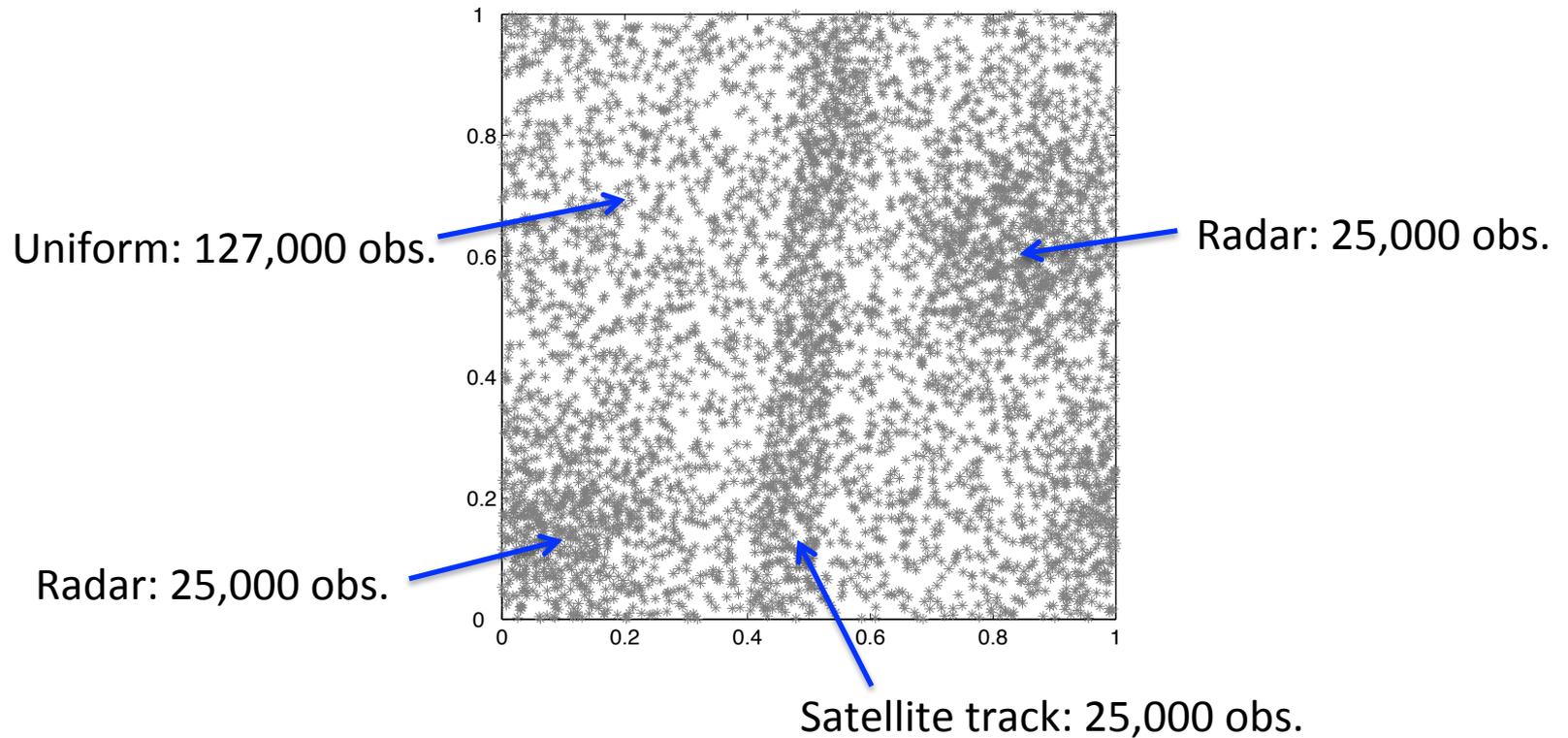
Sequential Ensemble Filter

7. Advance model to time of next observations.



Irregular Observations -> Load Balance Challenges

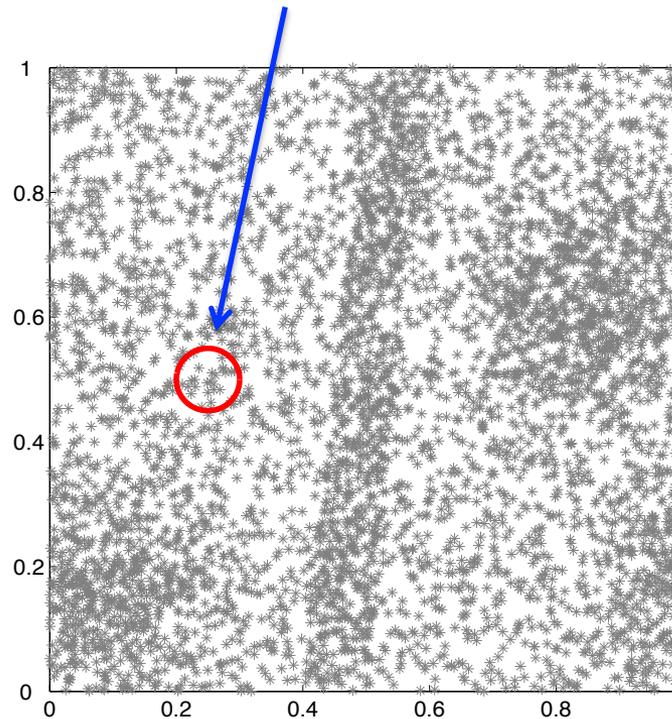
Simulate performance for idealized observation set (2% of obs shown).



Irregular Observations -> Load Balance Challenges

Simulate performance for idealized observation set (2% of obs shown).

Observations that are more than 0.05 apart are independent.



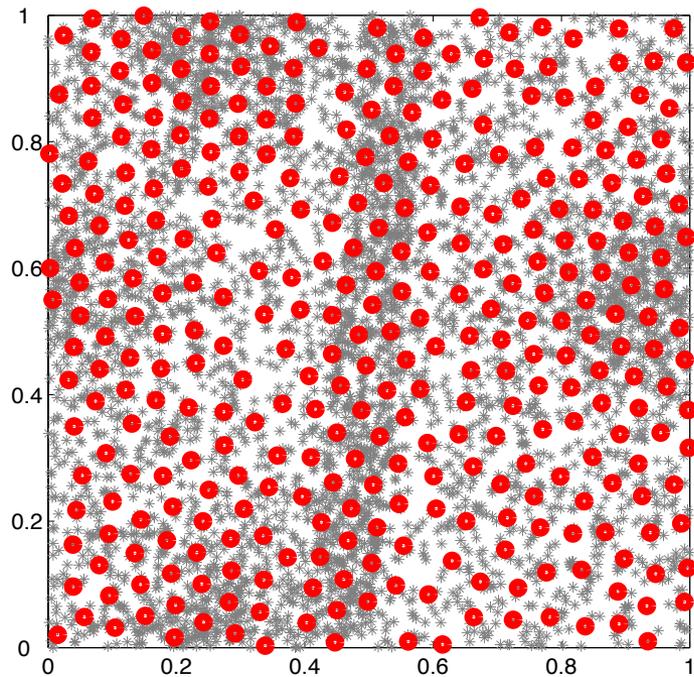
Parallel Observation Processing

- Find minimum number of subsets of independent observations.
- Mutual exclusion scheduling problem.
- Use greedy algorithm:
Decreasing Greedy Mutual Exclusion (DGME).

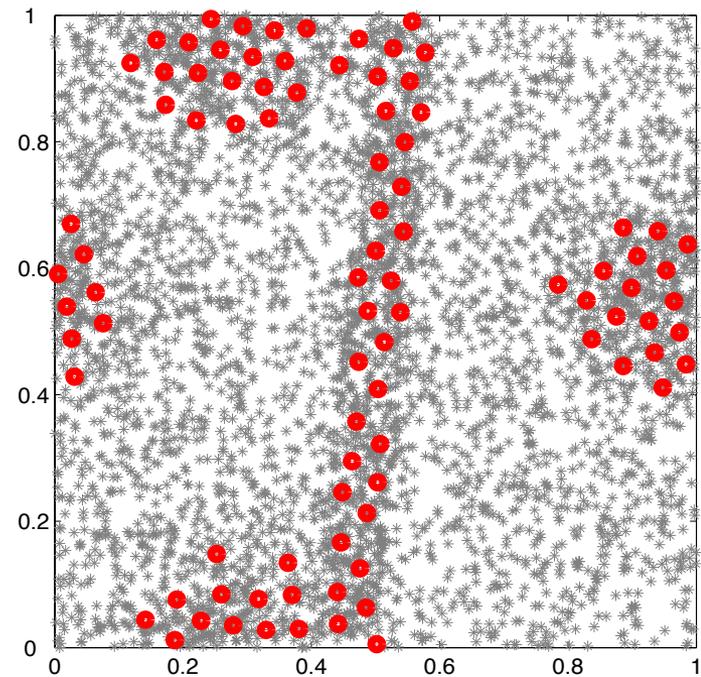
Irregular Observations -> Load Balance Challenges

Red shows observations in a given subset.

344 Observations in subset 1



91 Observations in subset 665

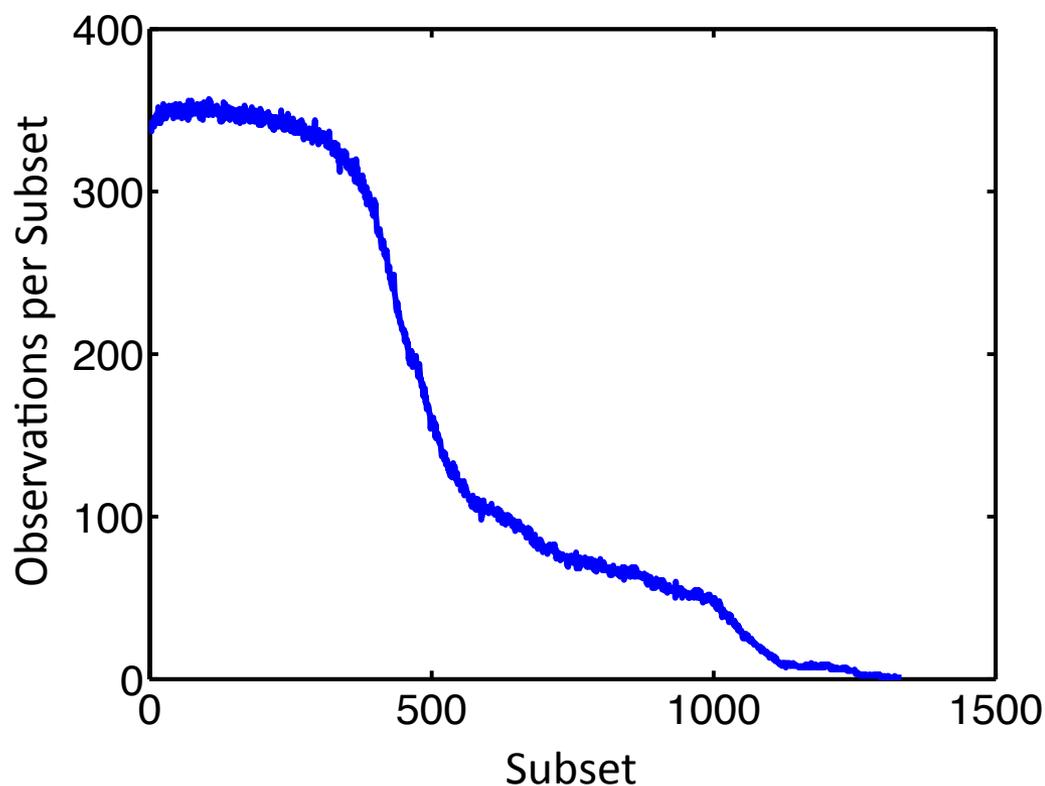


Irregular Observations -> Load Balance Challenges

Last subsets only have a few observations each.

These are in regions where satellite and radar overlapped.

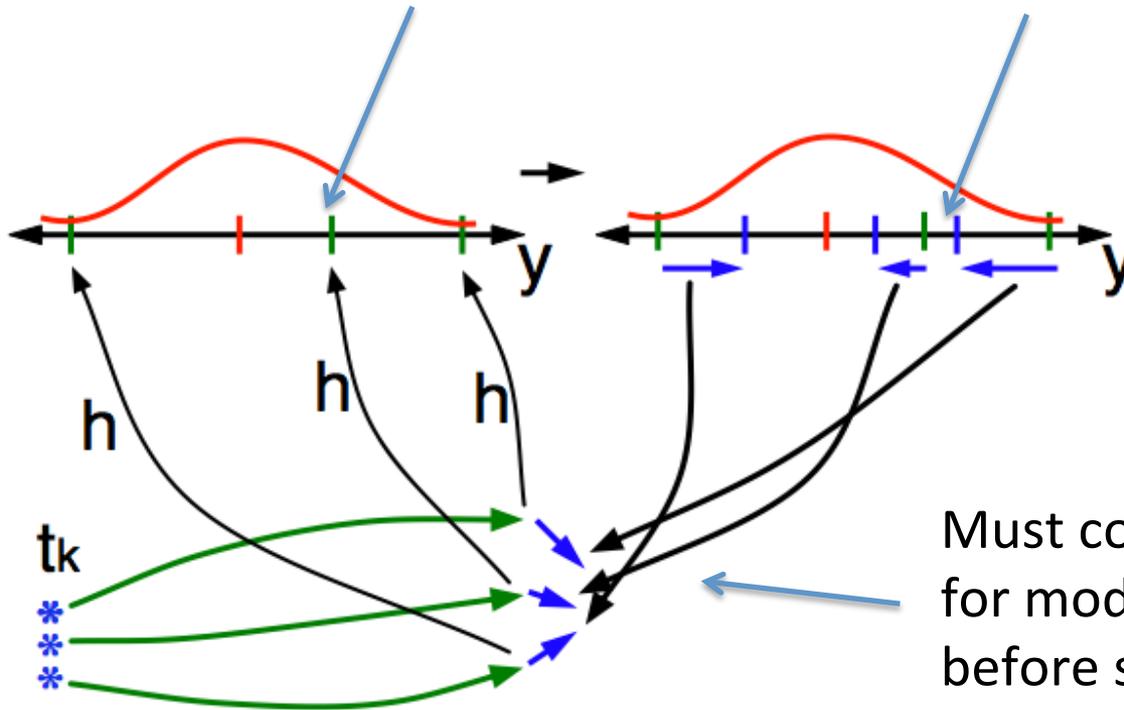
May be significant load balance issue.



Sequential Ensemble Filter

For each subset, can do following independently in parallel:

Compute forward operators, h Compute observation increments



Must compute all increments for model state variables before starting next subset.

Updating State Variables with a Subset's Observations

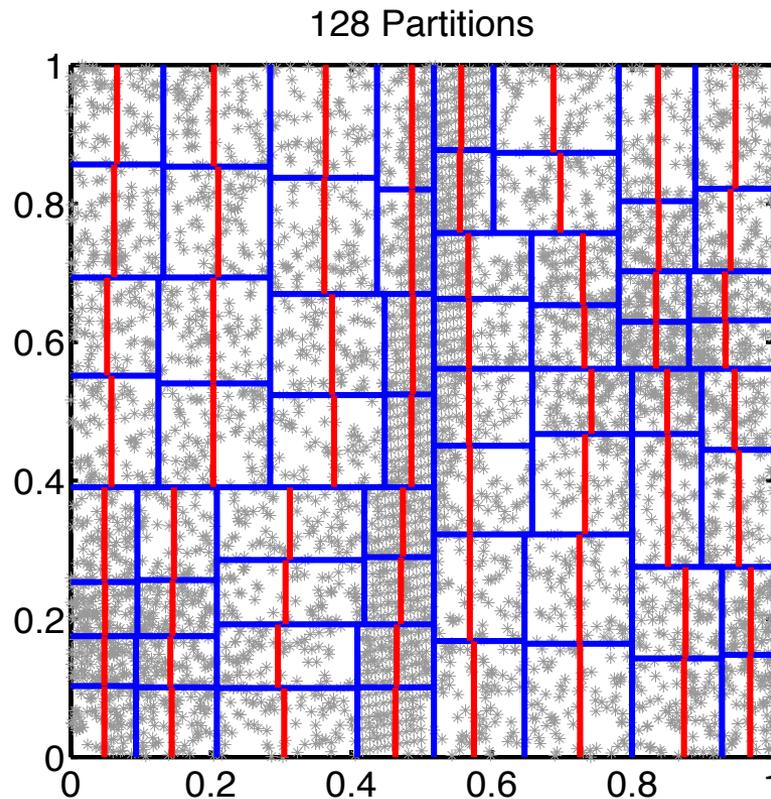
- Regress observation increments onto state ensemble.
- Distribution of state variables onto processes controls performance:
 - Load balance, slowest process controls speed,
 - Number of messages sent per observation.

Distribution of State Variables on $P=2^N$ Processes

- 1000x1000 grid of state variables.
- 3 possible distributions:
 1. Random: Each process gets random $1/P$ of state variables.
 2. Uniform: Divide domain into P squares of same size.
 3. Balanced: Total work for all observations approximately equal.

Balanced Rectangular Partitions for 1000x1000 grid points

Partition each rectangle so sum of total work on each side is nearly equal.



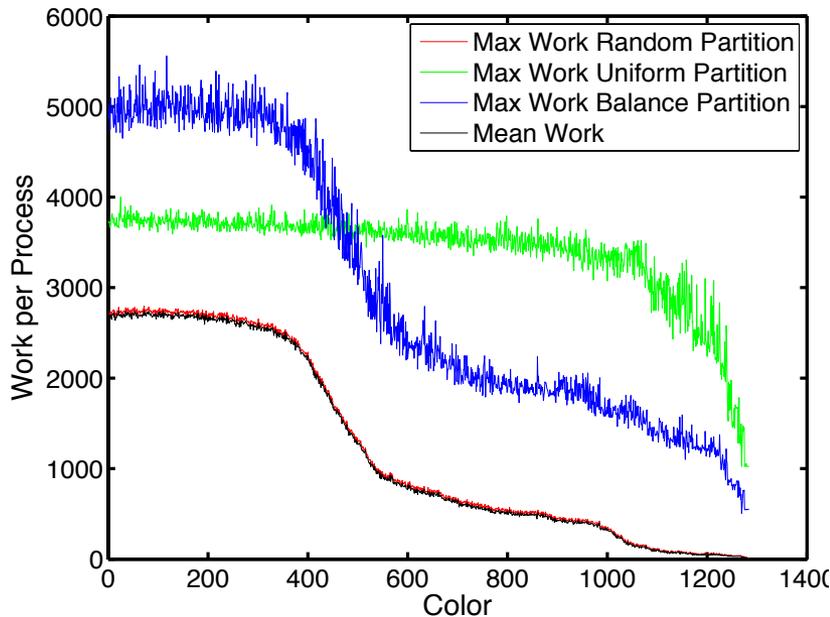
Load Balance for Different State Variable Distributions

Random always best.

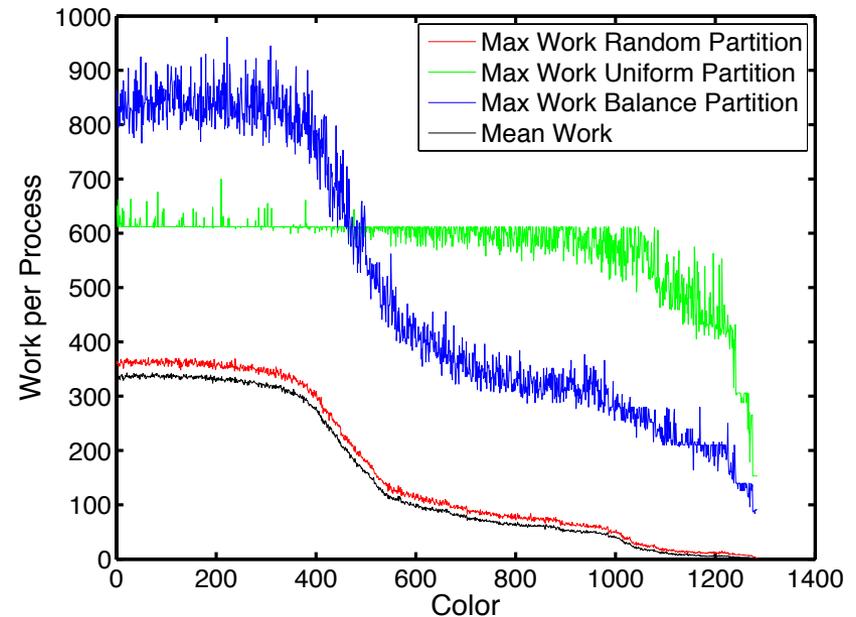
Uniform better than Balanced for early observation subsets.

Balanced better than Uniform for later observation subsets.

1024 Processes

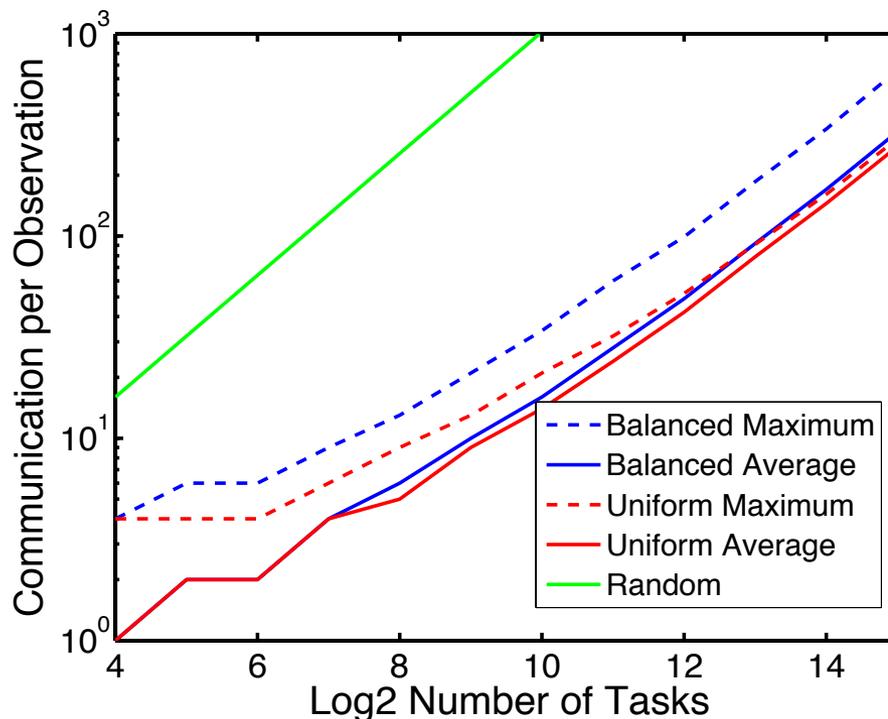


8192 Processes



Communication Cost for Different State Variable Distributions

Total Number of Messages for Each Observation



Balanced slightly more than Uniform.

Random requires a message to EVERY process for each observation! Very costly.

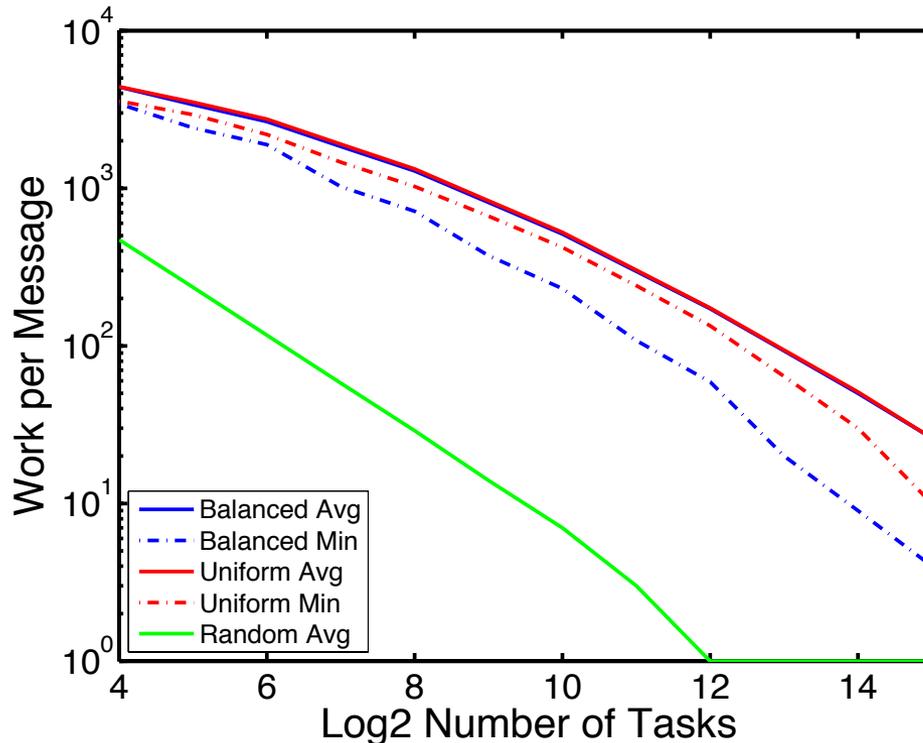
Making Effective Use of Coprocessors

- Many fast, cheap processors available for each process:
 - GPUs,
 - Intel Phi.
- Communication to coprocessors (even from local memory) is slow.
 - Getting a message from off-processor can be really slow.

Making Effective Use of Coprocessors

For efficient use, need lots of computation per communication.

Look at number of state variable updates per received observation increment.



Balanced, Uniform similar.

Scales well to many processes.

Optimistic about making efficient use of coprocessors.

Random has much less work per communication, probably won't work.

Conclusions

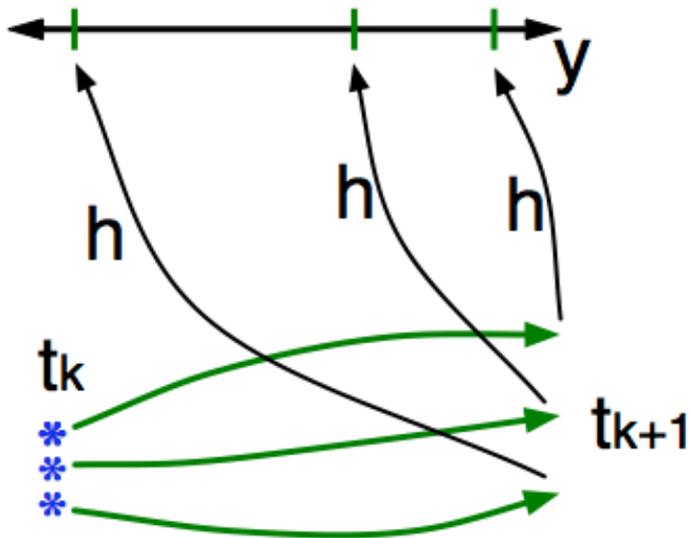
- Trade-offs between computation and communication efficiency.
- Careful assignment of state variables to processes crucial.
- May be able to use coprocessors, even for large process counts.
- We are interested in collaborations on tools or techniques.

Learn more about DART at:

<http://www.image.ucar.edu/DAReS/DART>

Computing Forward Operators

- Forward operator h is an arbitrary function of state variables
- Often 'local' function of state (interpolation)
- Can be complex and non-local (radio occultation ray tracing)



Computing Forward Operators

- We use MPI-2 non-blocking remote memory direct access
- One process computes h for all ensemble members
- Grabs any required state variables from other processes
- Number of messages minimized if adjacent state is on same process
- Communication volume minimized if local state is on process computing h

What Matters when Selecting State Variable Distribution?

- Relative cost of computation versus communication (machine dependent)
- Relative density of state variables and observations
- Observation density (number of observations close to given observation)
- Amount of observation spatial heterogeneity

- DART will implement all three distributions discussed here
Also tools for implementing arbitrary distributions

- More focused efforts can probably pick one

Other Optimizations we are exploring

1. More state variable partitions than processes
 - For balanced partition, make kP partitions where P is number of processes, k is a small integer
 - A process gets partitions with both large and small numbers of observations
 - Large partitions have more work for early colors
 - Small partitions have more work for late colors
 - Mix can have better load balance thru time

Other Optimizations (II)

2. Pre-compute observations for more than one color

- Theory allows computing h for any number of obs at one time
- Observations from later colors are updated just like state variables
- Can 'smooth' out load for state increments
- Extra computation for observations (may be small)
- Current DART computes ALL observations at once

3. Divide observations into pieces, separate state variable partition for each

- Do uniform observations first, then satellite, etc.
- Increased communication to move state variables for different partitions