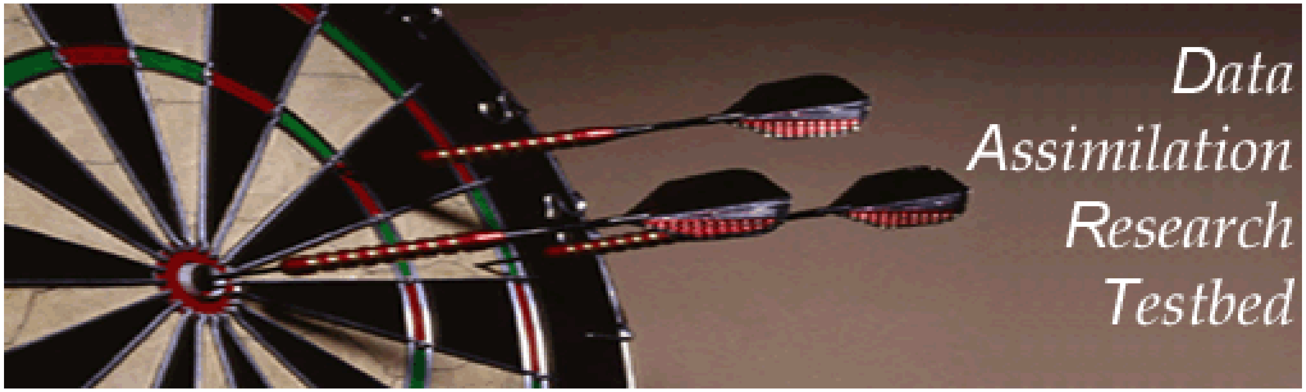
Data Assimilation Research Testbed Tutorial



Section 22: Parallel Algorithm Implementation

Version 2.0: November, 2006

A simple example assimilation cycle.

1. Model with six state vector components a, b, c, d, e and f.

2. Four observations at first time are p, q, r, and s.

3. Three threads referred to as PE0, PE1 and PE2.

4. Three ensemble members.

State vector elements referred to by a 2-character alphanumeric string.

First character is the state variable component.

Second character is which copy of this state vector component.

Copies can be ensemble members or other adjunct quantities.

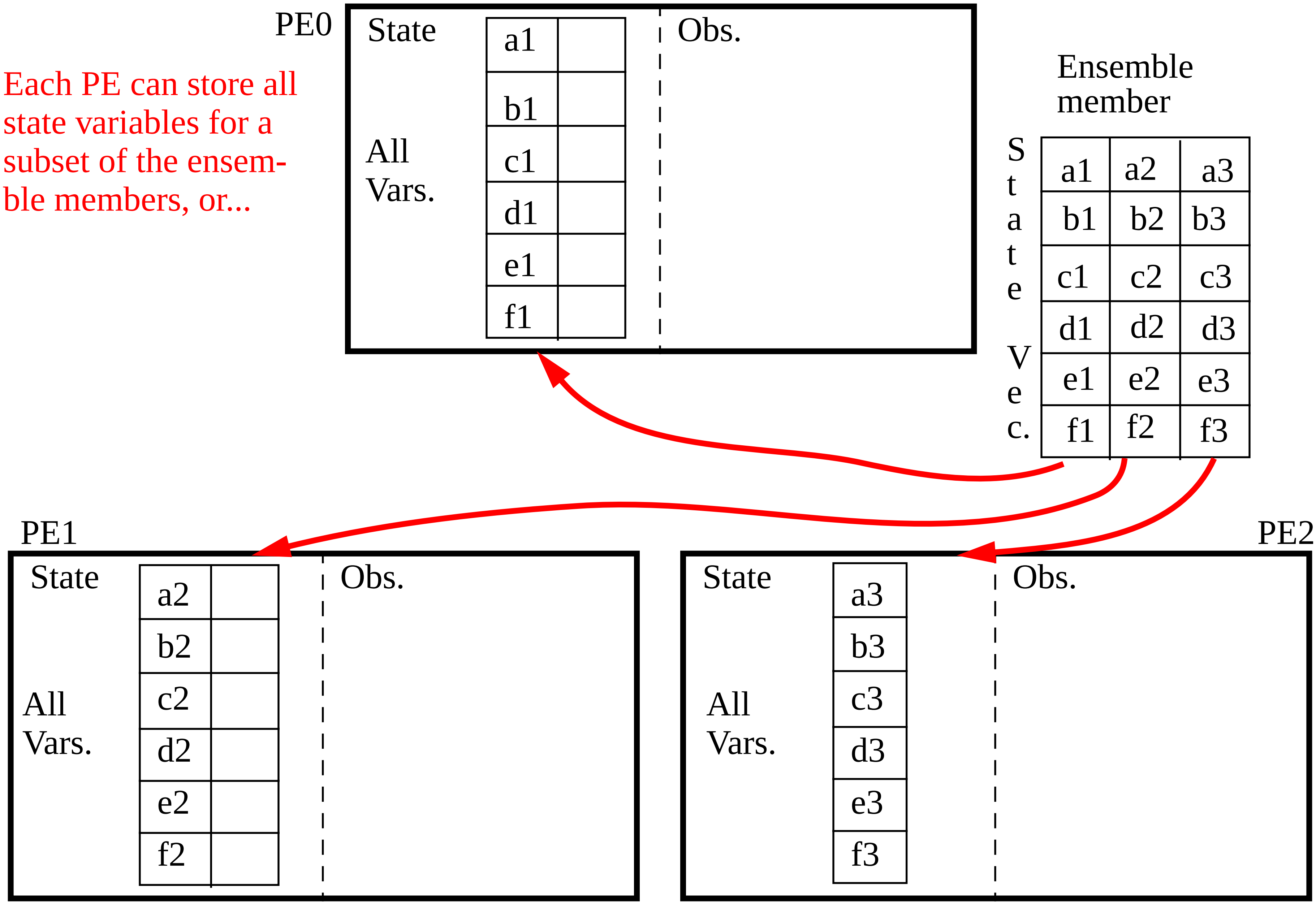
Complete list of copies for state vector:

1, 2, 3:	Ensemble members;
m:	Ensemble mean;
σ:	Ensemble standard deviation.

Example: b3 is the second state variable's third ensemble copy.

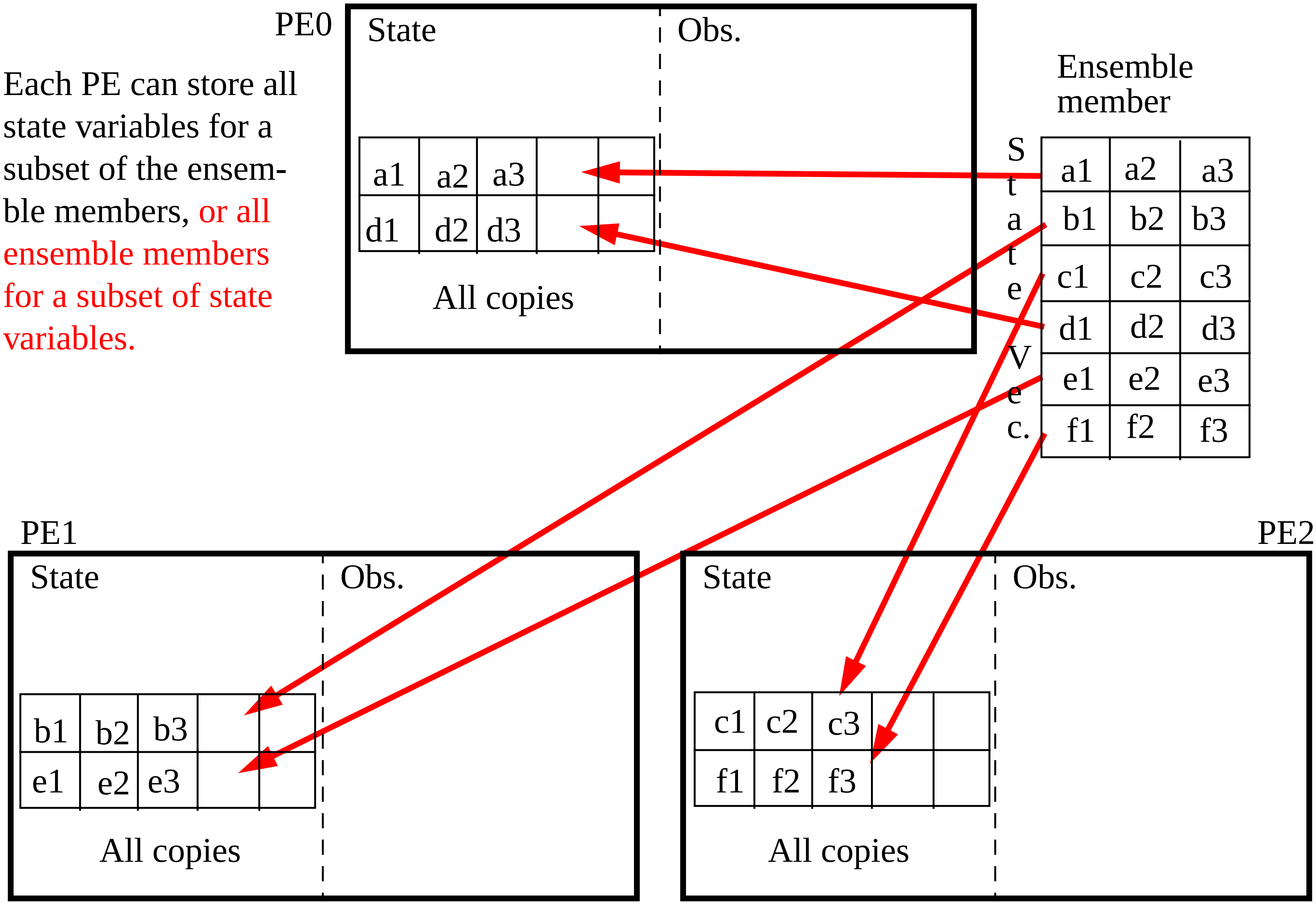
No single PE ever stores the complete state vector ensemble.

Each PE can store all state variables for a subset of the ensemble members, or...



No single PE ever stores the complete state vector ensemble.

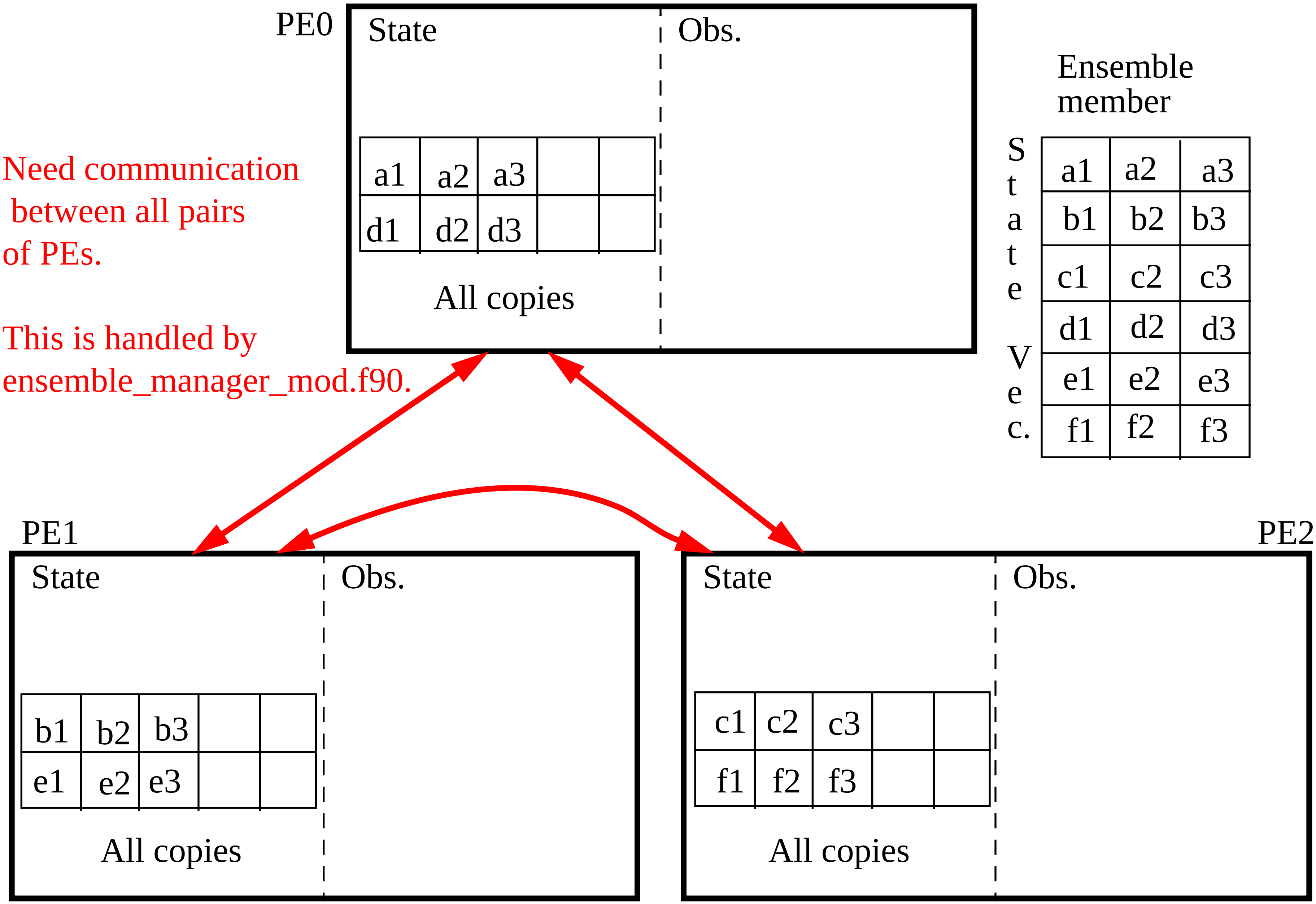
Each PE can store all state variables for a subset of the ensemble members, or all ensemble members for a subset of state variables.



Transition between two storage patterns is called a transpose.

Need communication between all pairs of PEs.

This is handled by ensemble_manager_mod.f90.



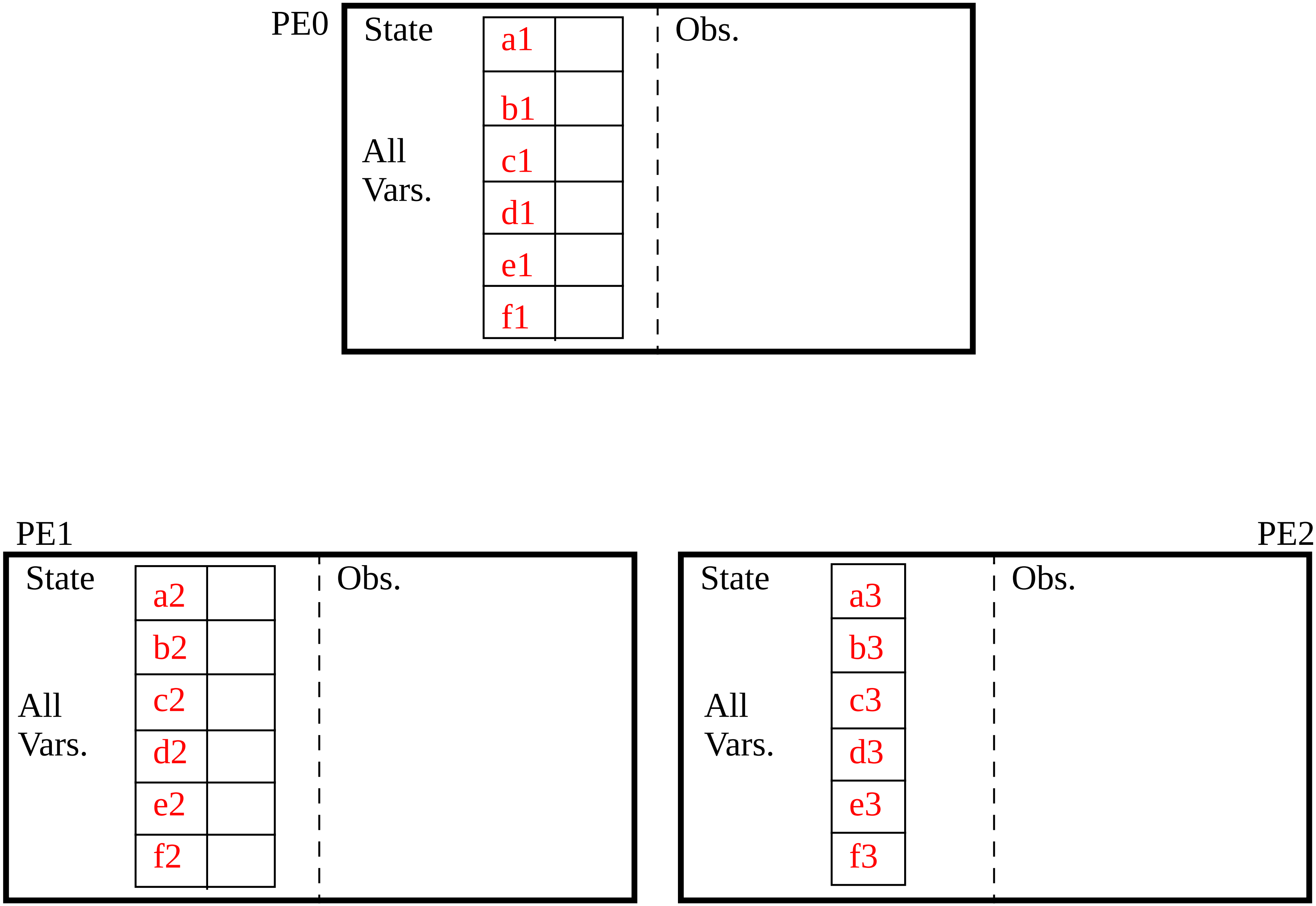
Also maintain copies of the observation priors for each observation.

Complete list of copies for observation priors:

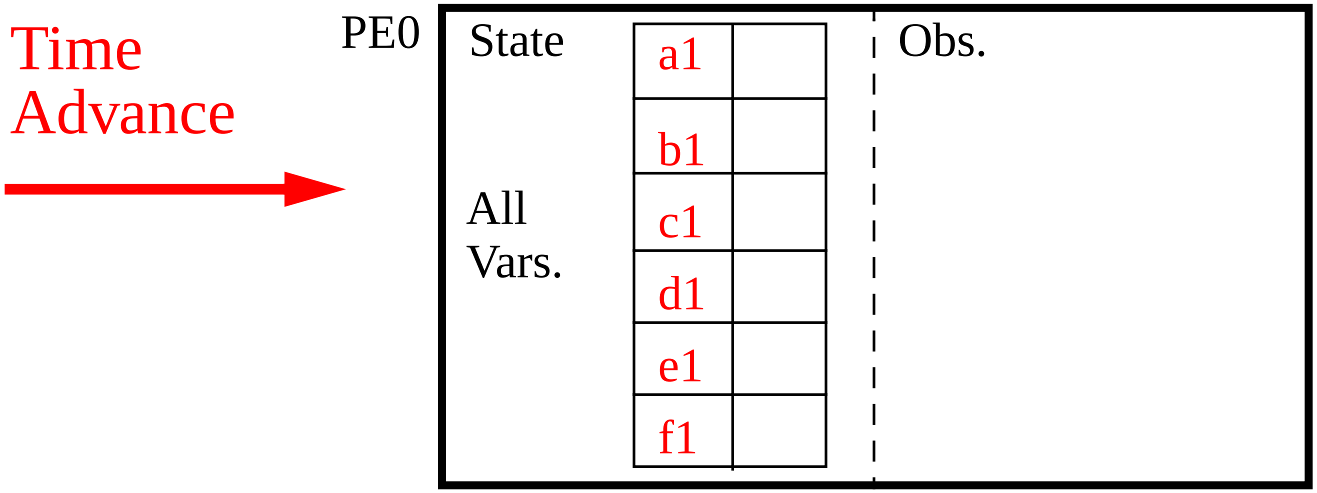
1, 2, 3:	Ensemble member;
k:	Unique integer index from obs_sequence file;
o:	Observed value from the obs_sequence file;
Σ:	Obs. error variance from the obs_sequence file;
Q:	The DART quality control value;
m:	Ensemble mean;
σ:	Ensemble standard deviation;

Examples: q1: second observation, 1st ensemble copy;
rk: integer index for 3rd observation;
pm: Ensemble mean of 1st observation.

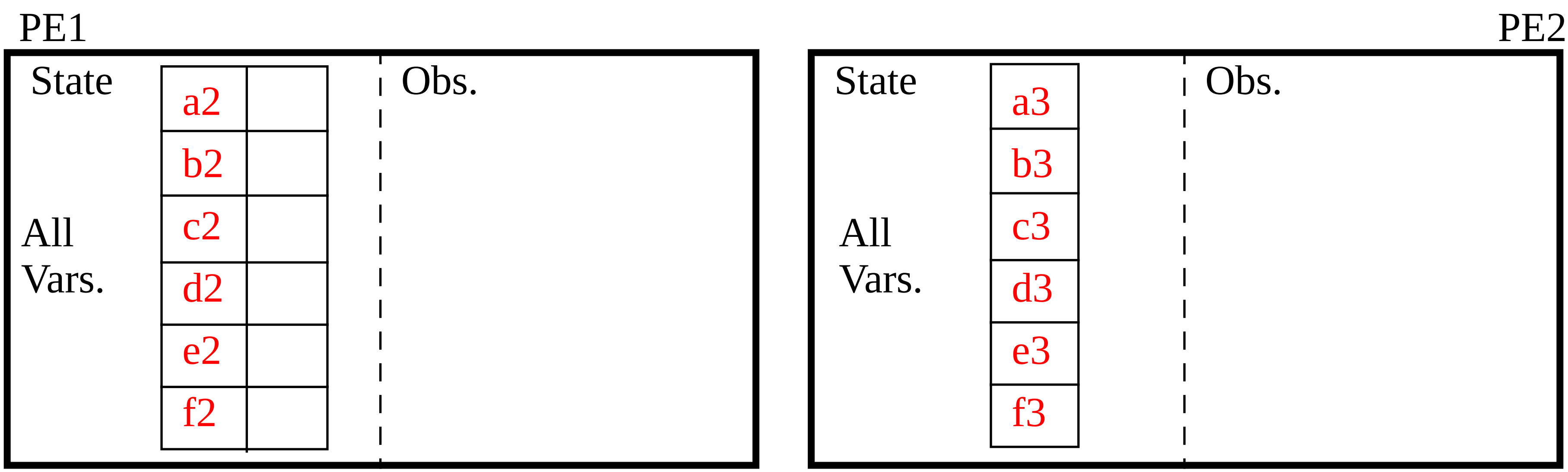
Each PE starts with all state variables for a subset of ensemble members.



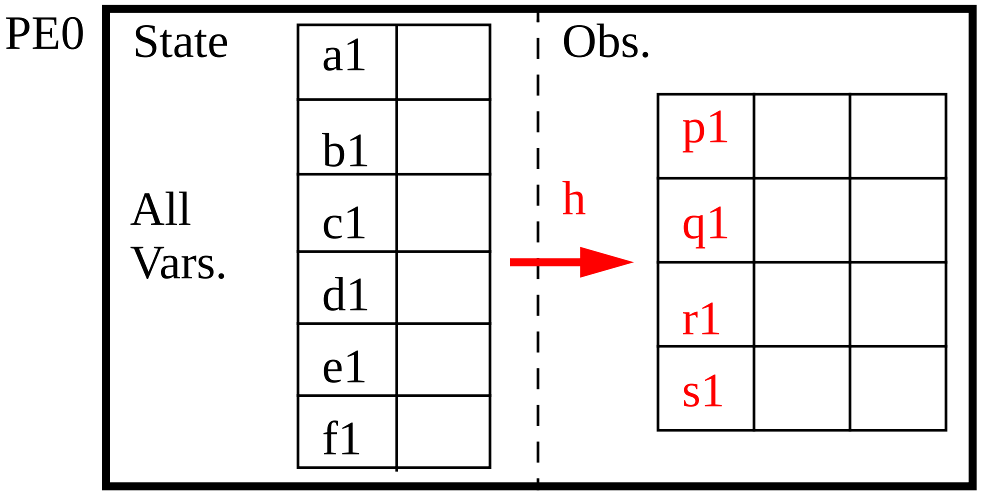
Advance state vectors in time.



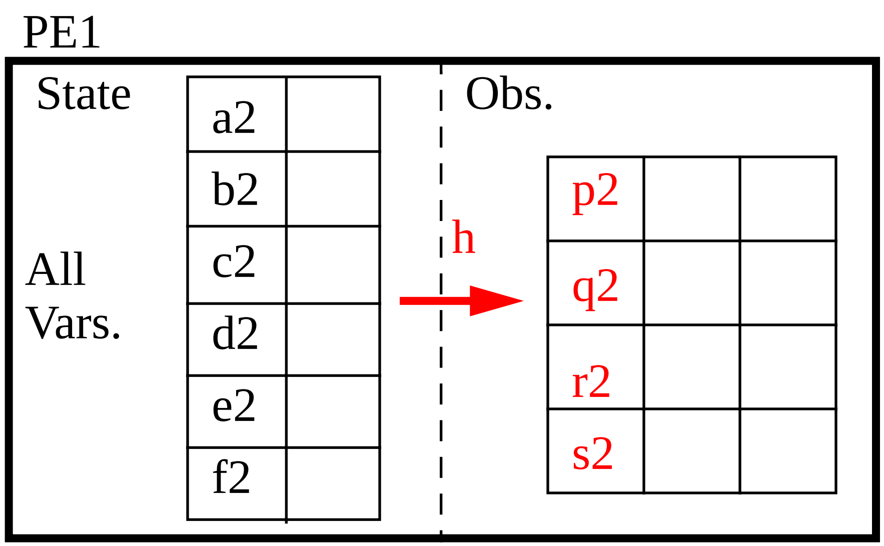
Each PE advances its ensemble members to the next observation time.
Here, each PE advances 1 member (see move_ahead, obs_model_mod.f90).
With more ensemble members than PEs, PEs would advance more than 1.

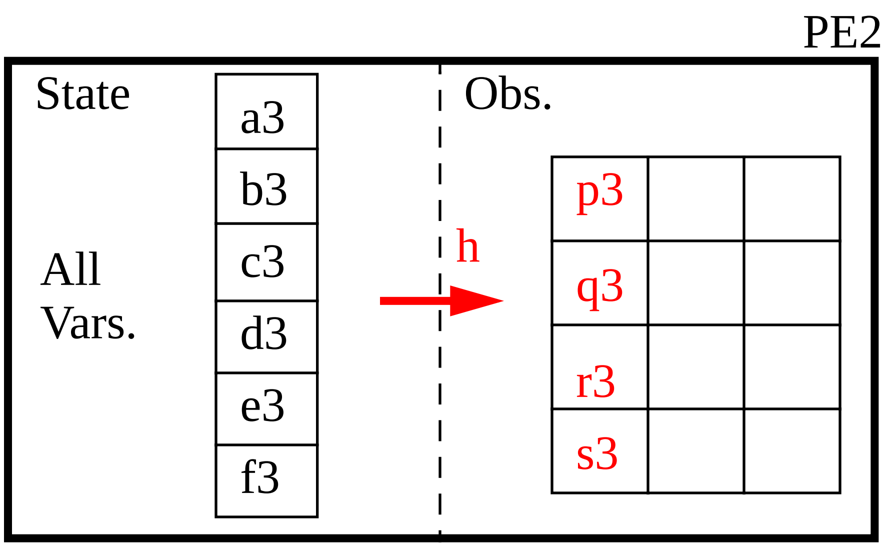


Apply forward observation operators.



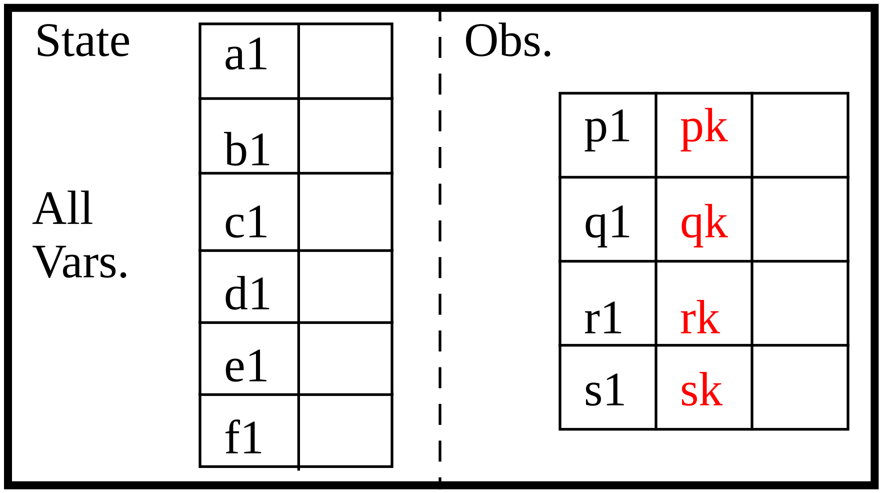
Each PE computes forward observation operators for its ensembles.
Here, each only does one set of forward ops. (see get_obs_ens in filter.f90)
Four observations (p, q, r, s) are available at this time.





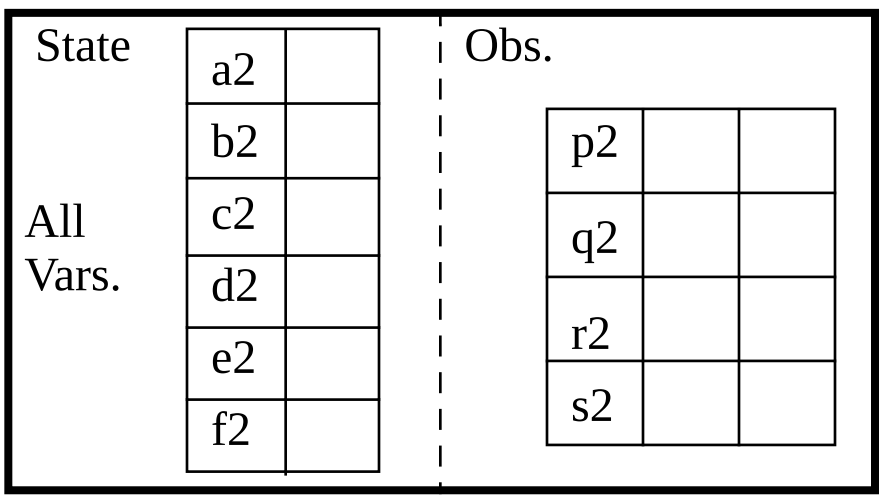
Attach unique integer key from observation sequence.

PE0

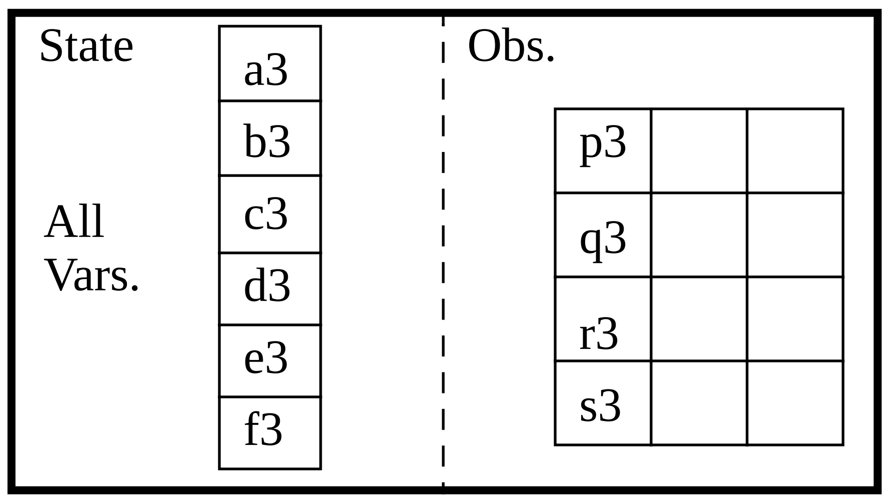


Each observation has a unique integer key.
It is stored like an additional ensemble member (but not treated like one).
The key comes from the obs_sequence data structure.

PE1



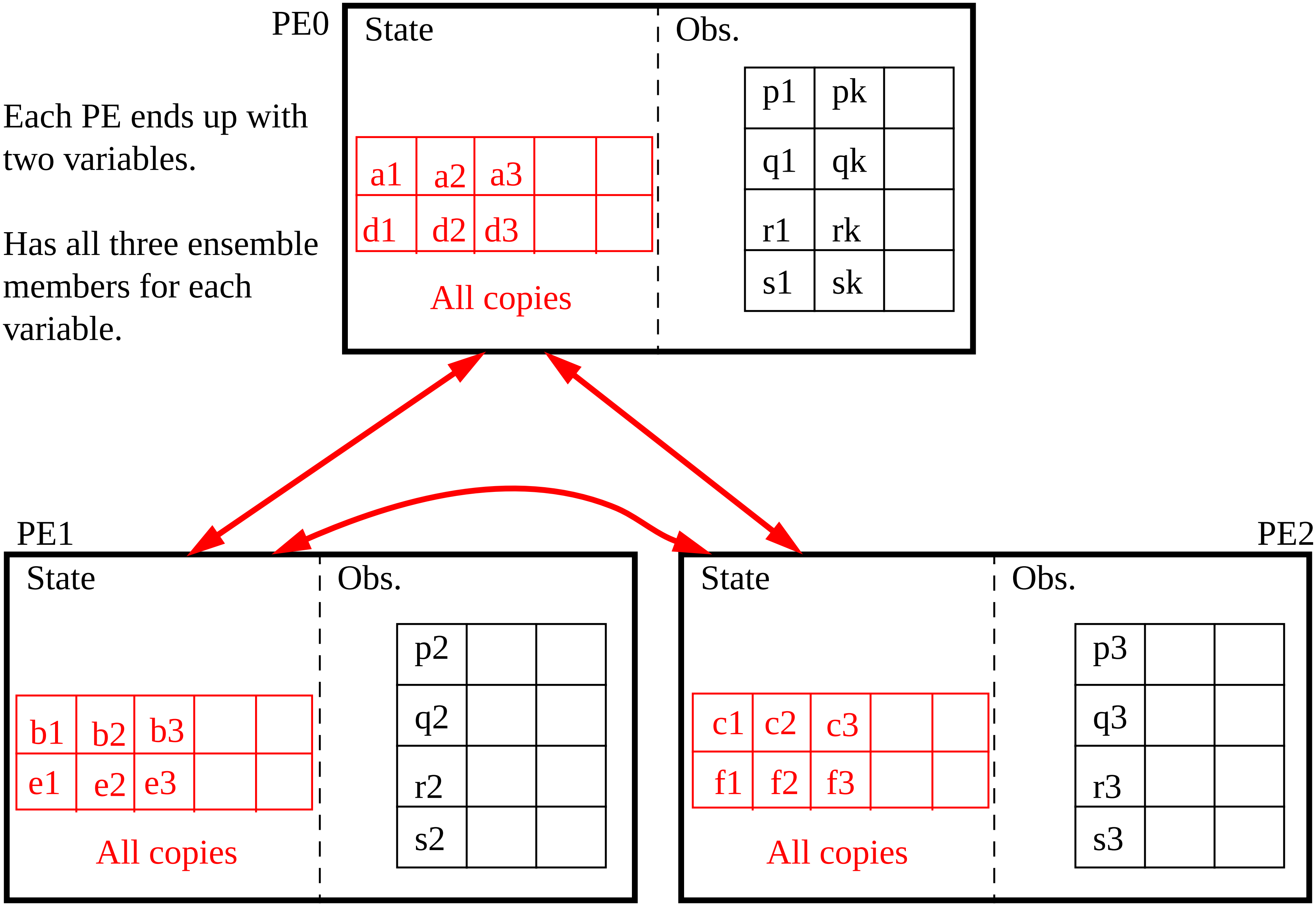
PE2



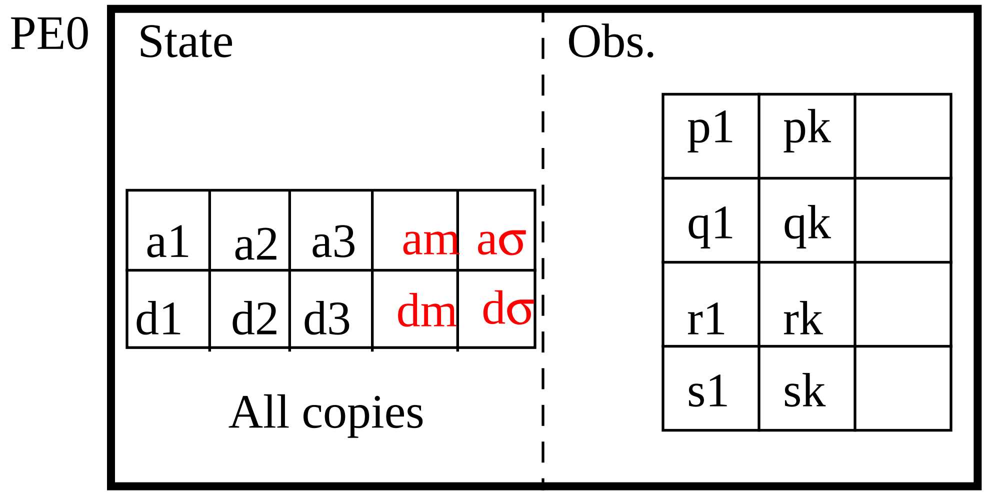
Transpose state vectors to have all copies of subset of state variables.

Each PE ends up with two variables.

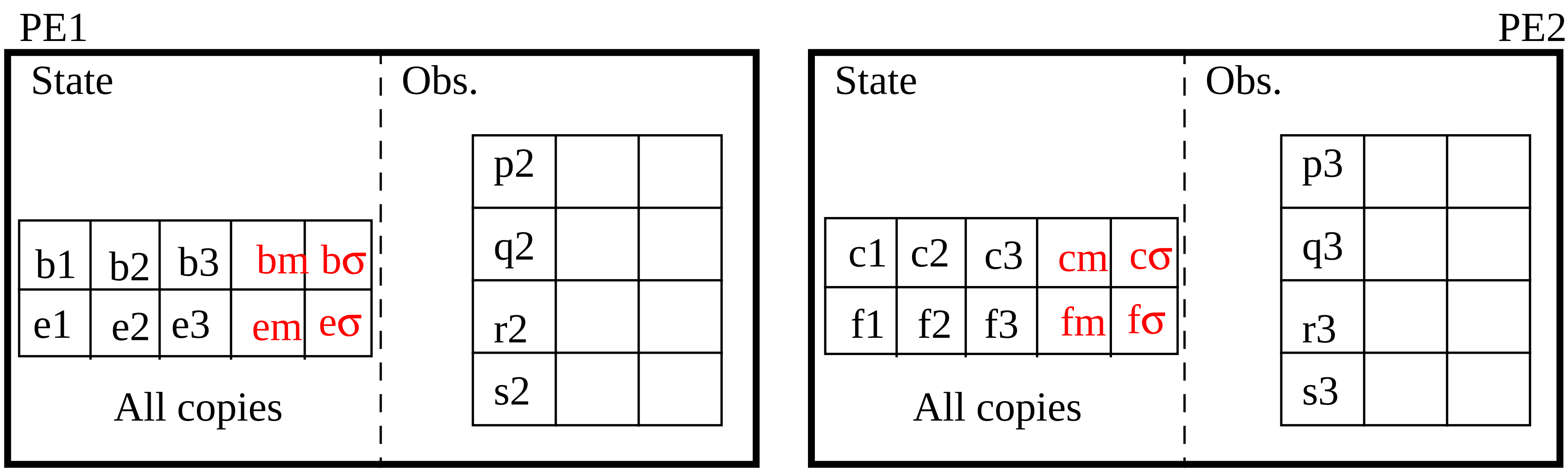
Has all three ensemble members for each variable.



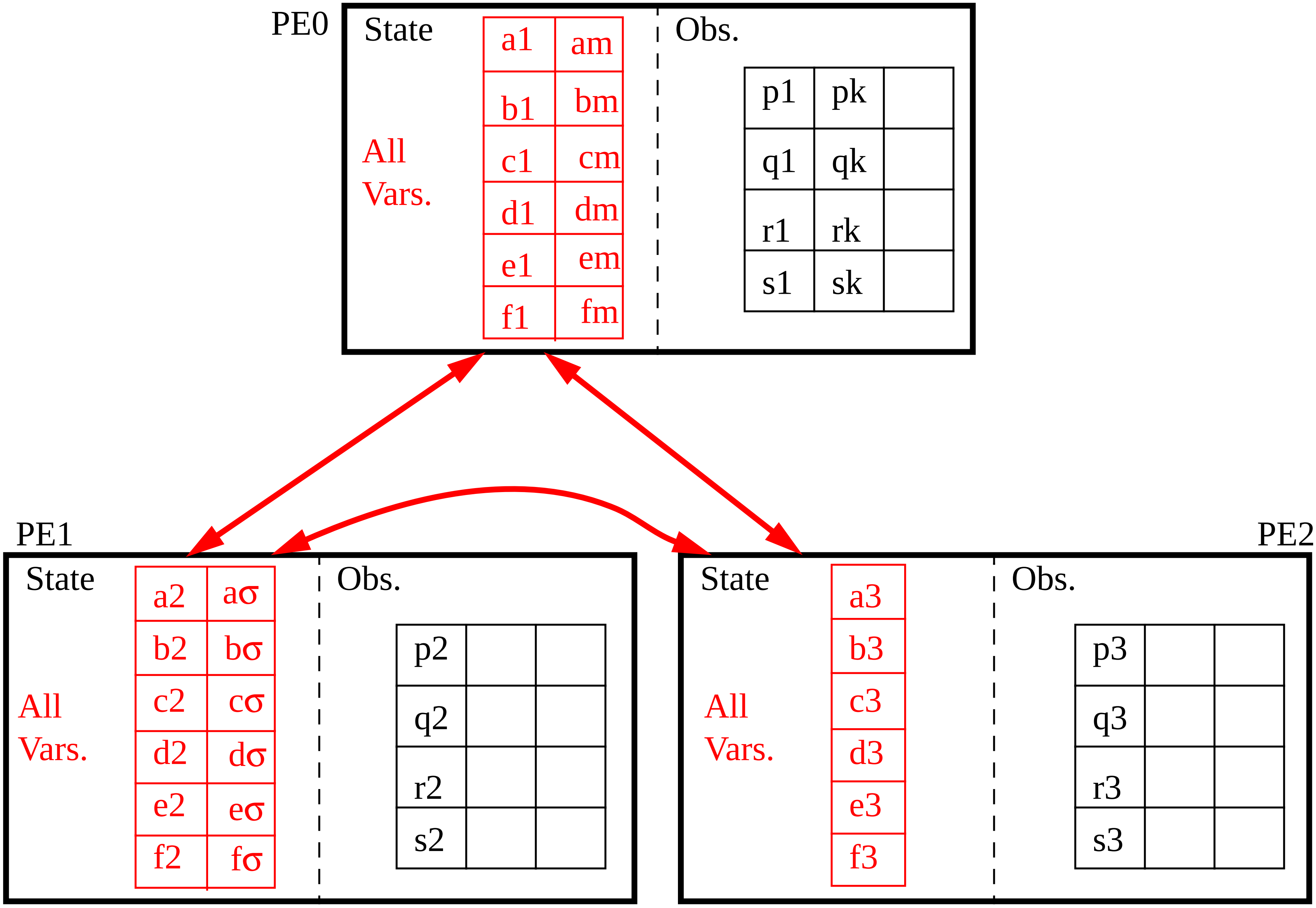
Compute ensemble mean and standard deviation.



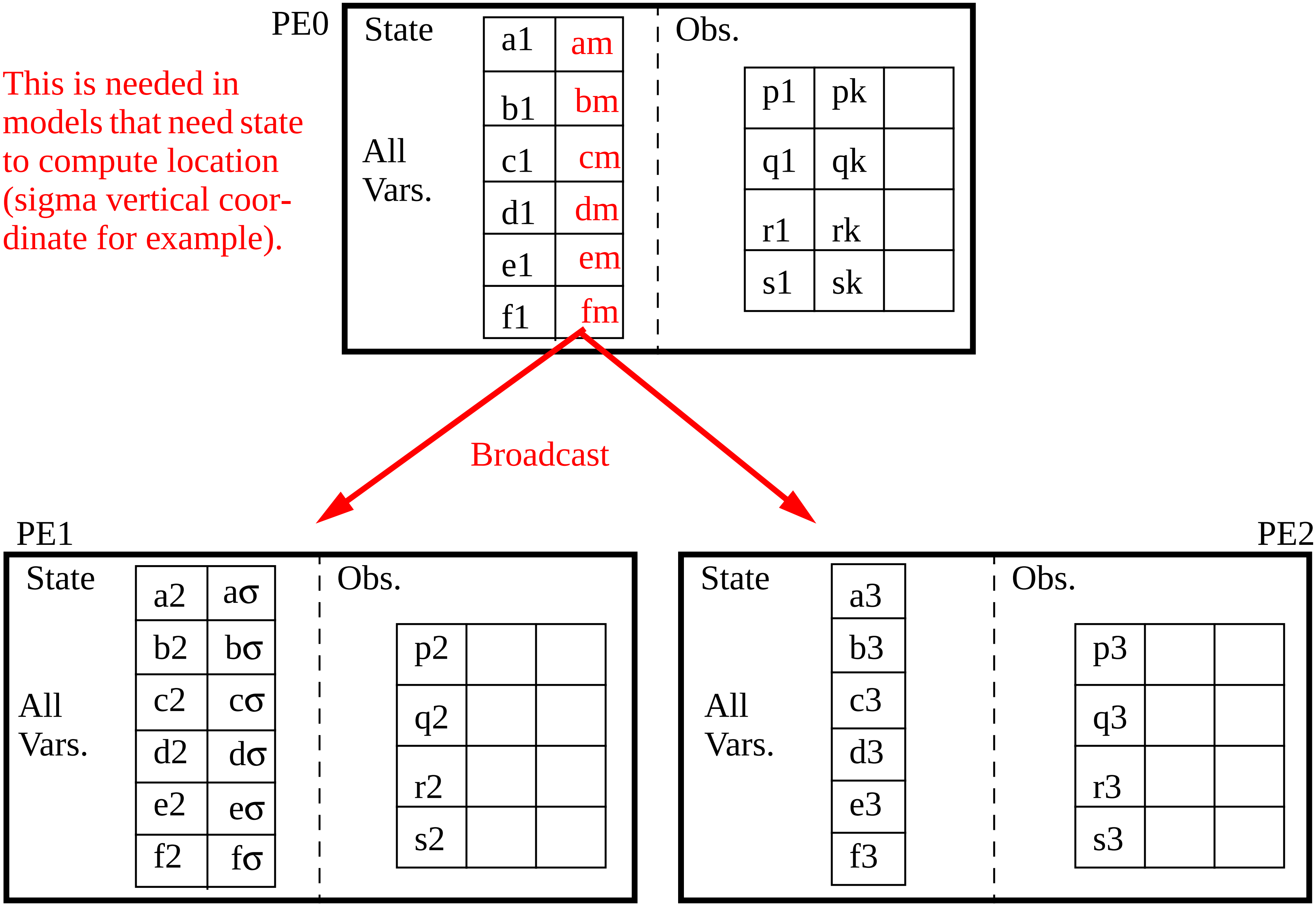
Each PE computes mean and standard deviation for its state variables.
These are stored as if they are extra ensemble copies.



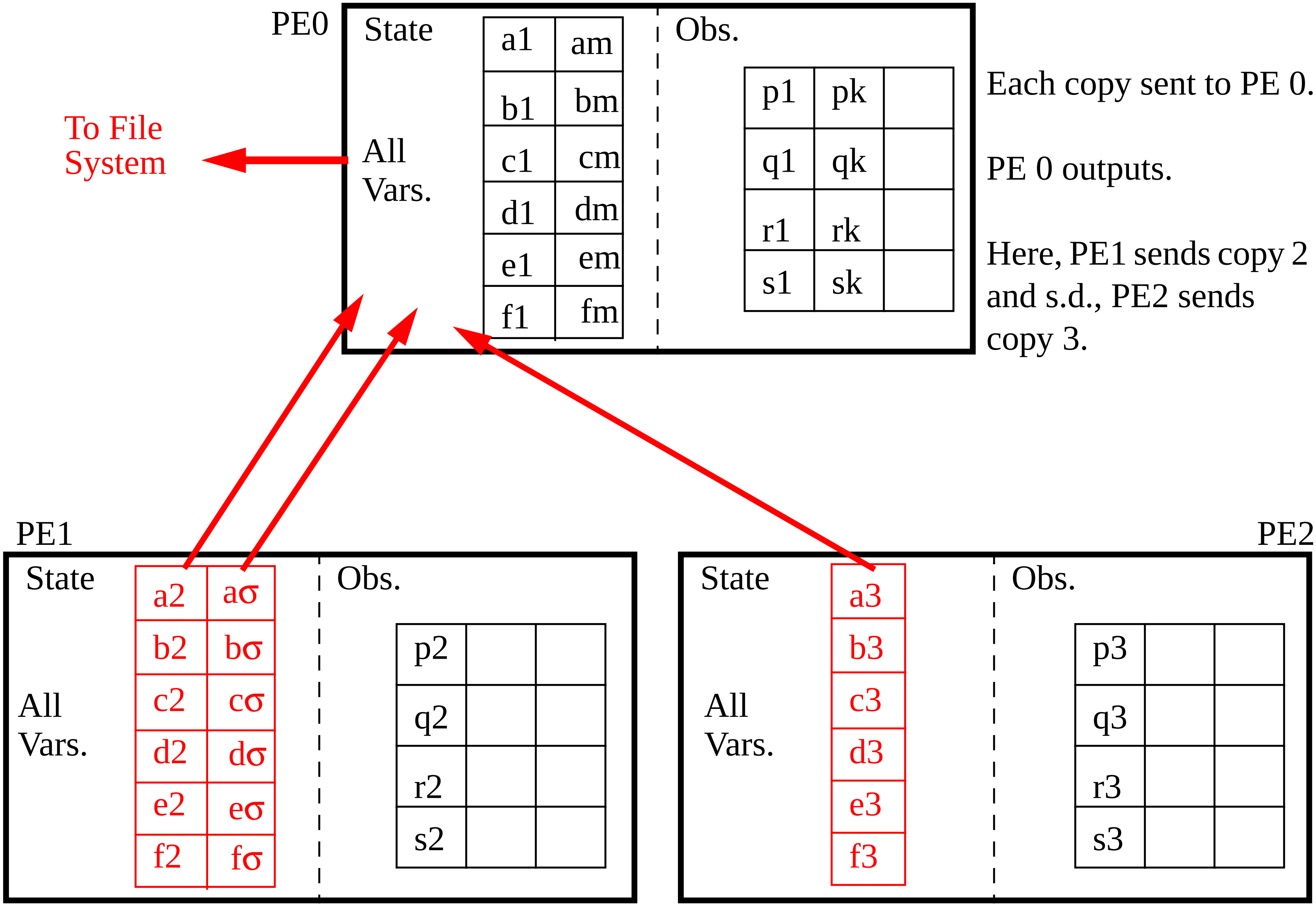
Transpose back to subset of copies, all variables for state.



Can now send ensemble mean state vector to each PE for later use in location computation.



Prior state space diagnostics are now output.

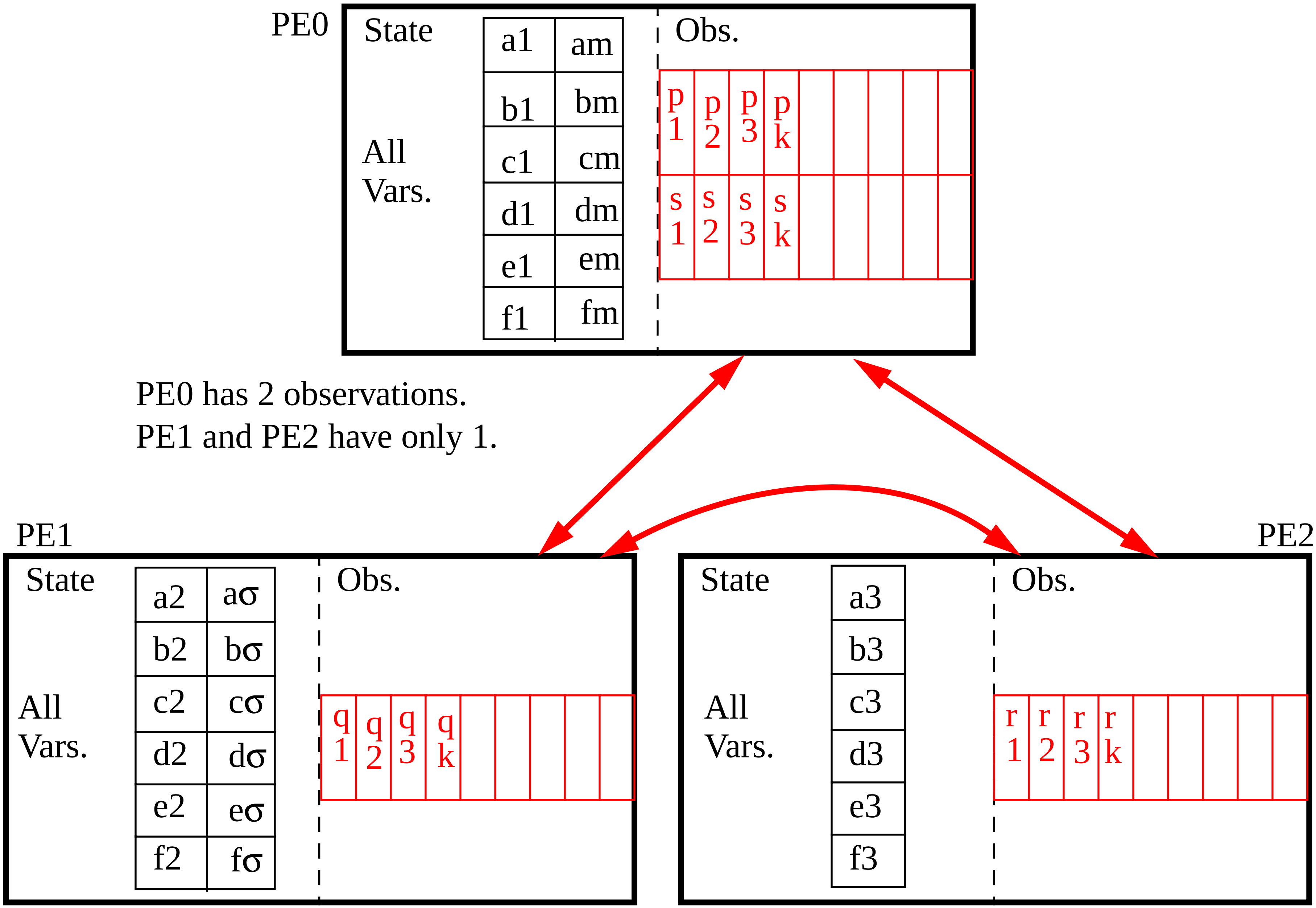


Each copy sent to PE 0.

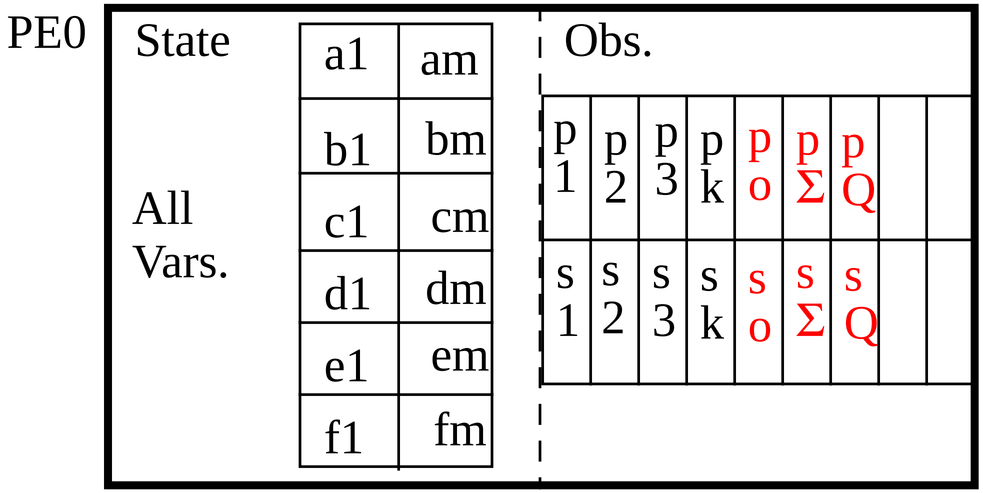
PE 0 outputs.

Here, PE1 sends copy 2 and s.d., PE2 sends copy 3.

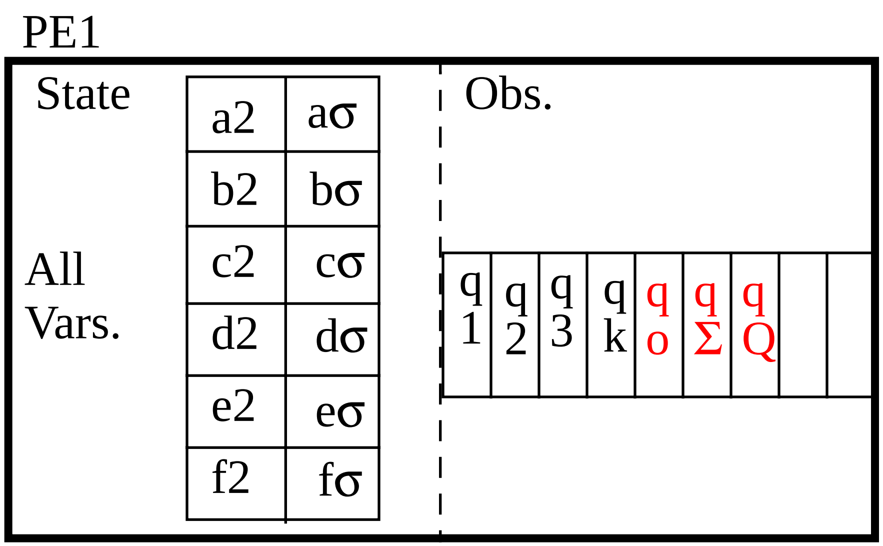
Transpose observations to all copies of subset of observations.

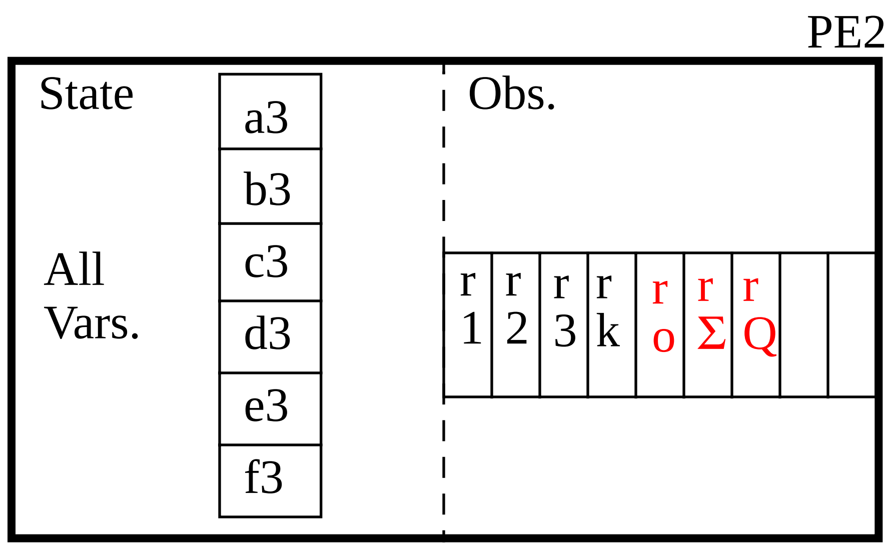


The observation value, observation error variance, and QC are added in.

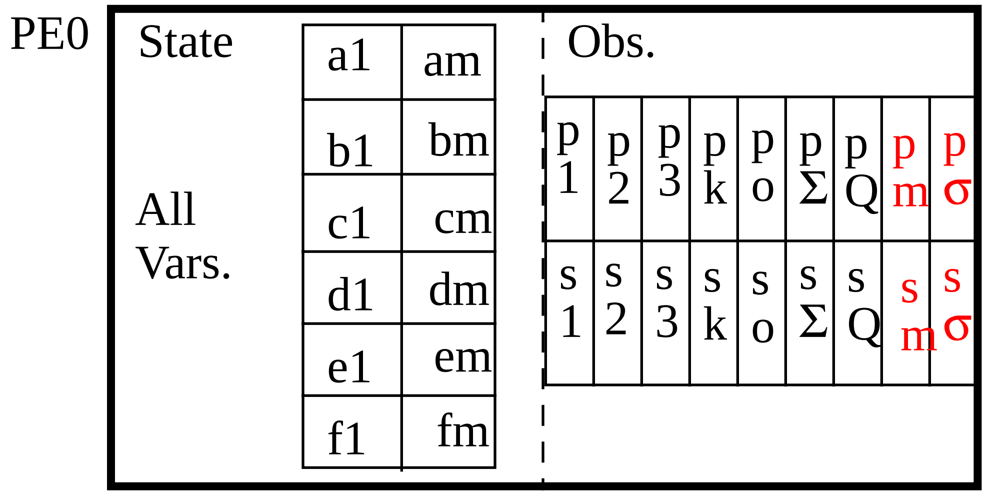


The value and observation error variance are from the obs_sequence file.
The QC is a merge of prior QC (NCEP) plus any failed forward obs. operators.



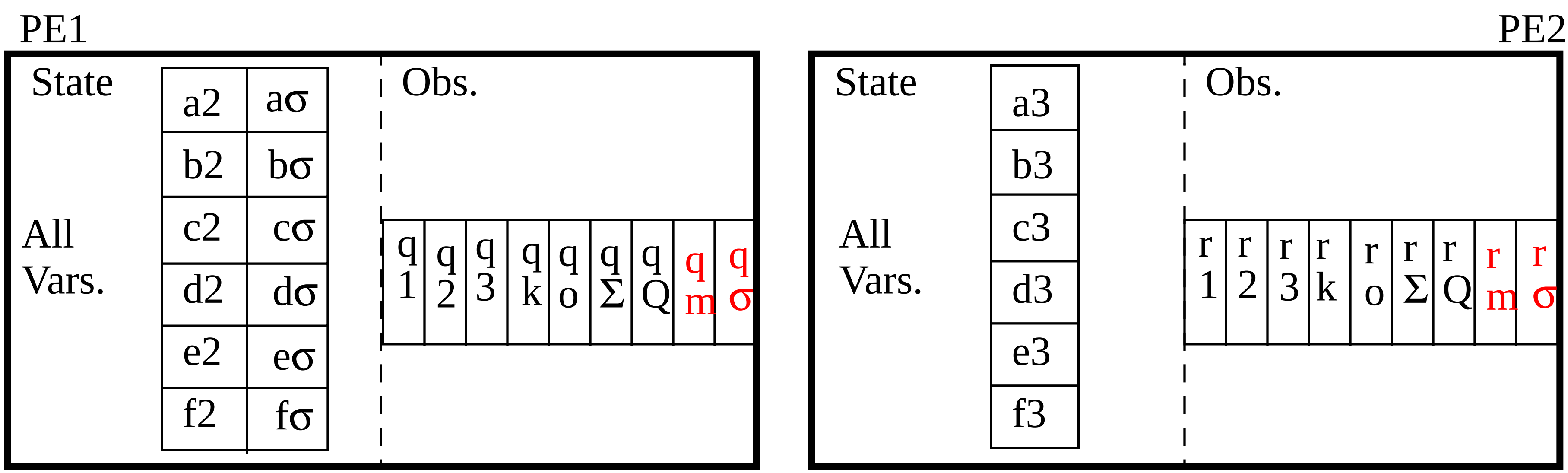


The ensemble mean and prior observation standard deviation are computed.

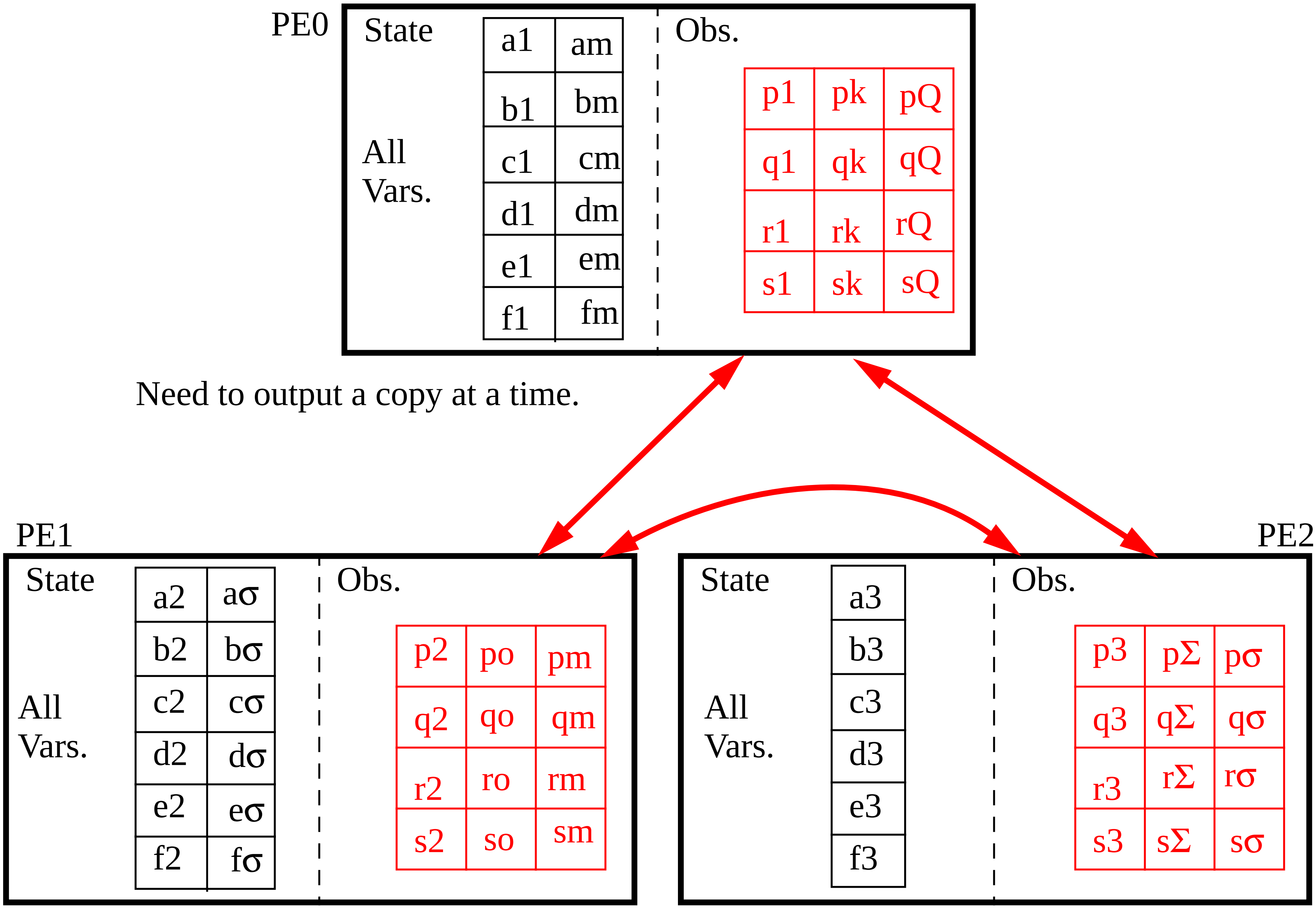


Each PE computes mean and s.d. for its observations.

End up with a total of 6 extra copies (like ensemble members) for each obs.



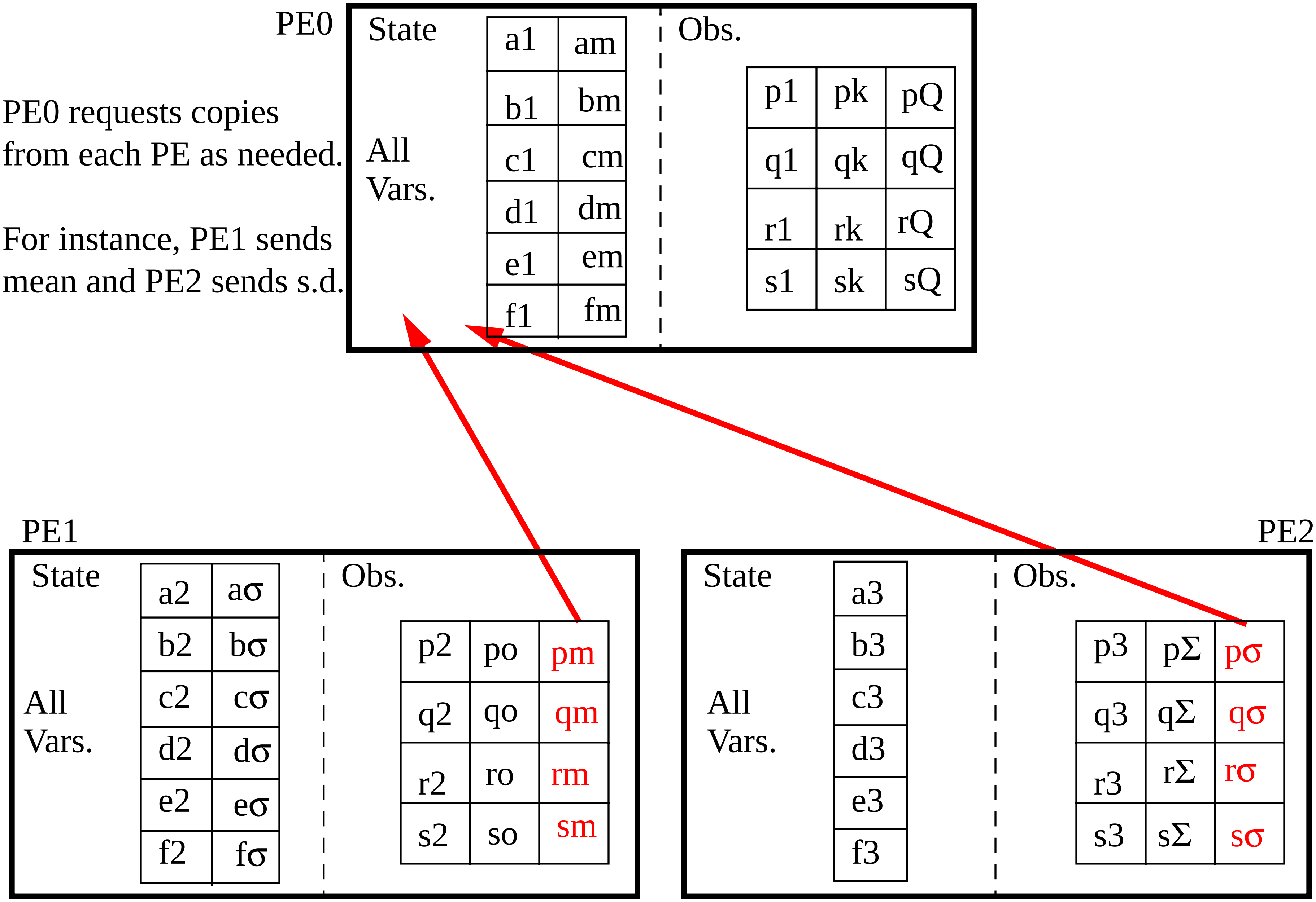
Transpose observations back to subset of copies of all observations.



PE0 outputs all requested copies sequentially.

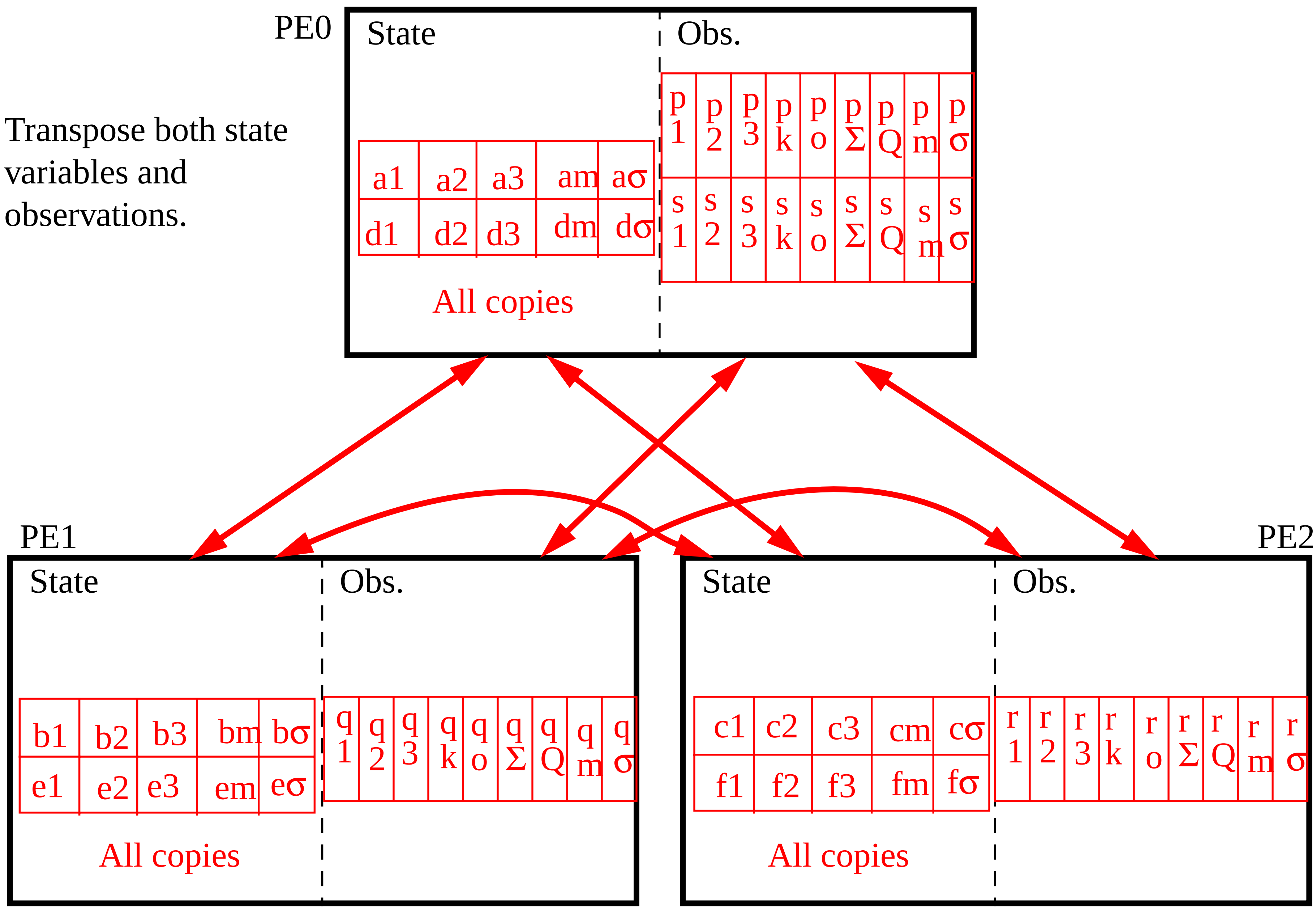
PE0 requests copies
from each PE as needed.

For instance, PE1 sends
mean and PE2 sends s.d.

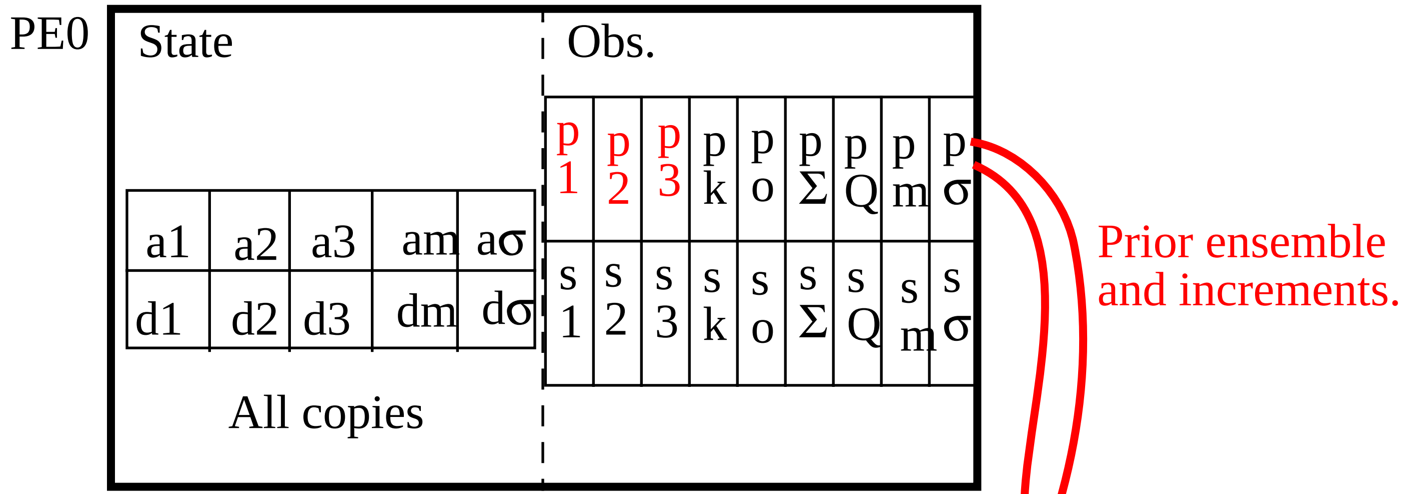


Need all copies of subset of variables and observations for assimilation step.

Transpose both state variables and observations.



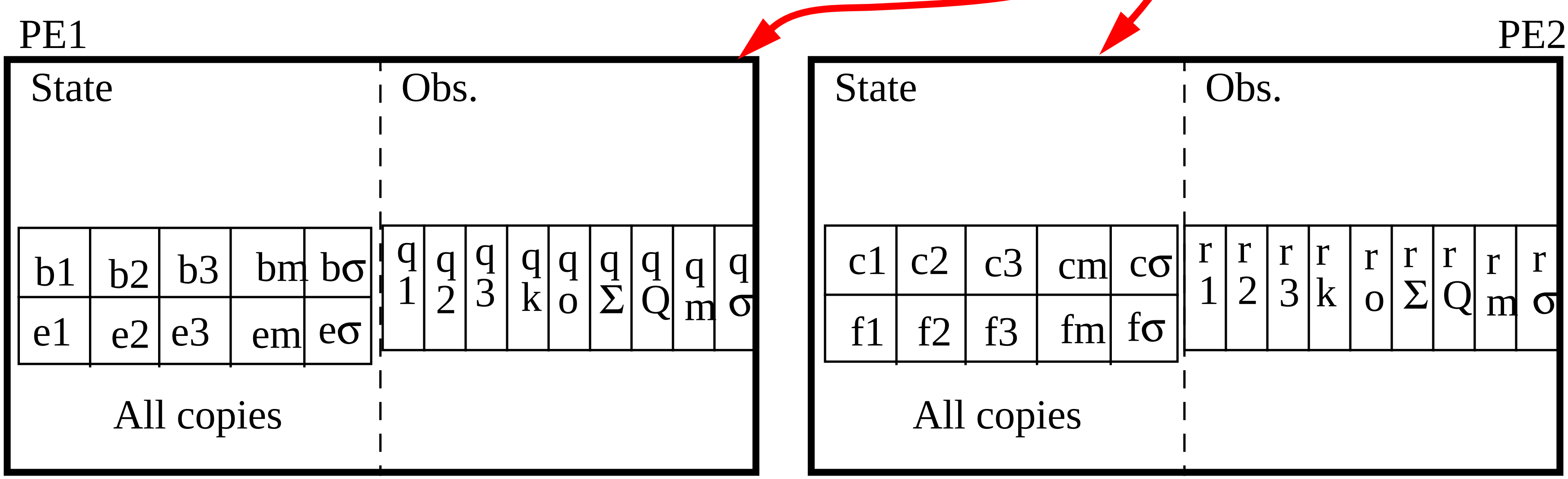
Call to filter_assim in assim_tools_mod.f90 to do the assimilation of p, q, r and s.



PE0 has the first observation, p.

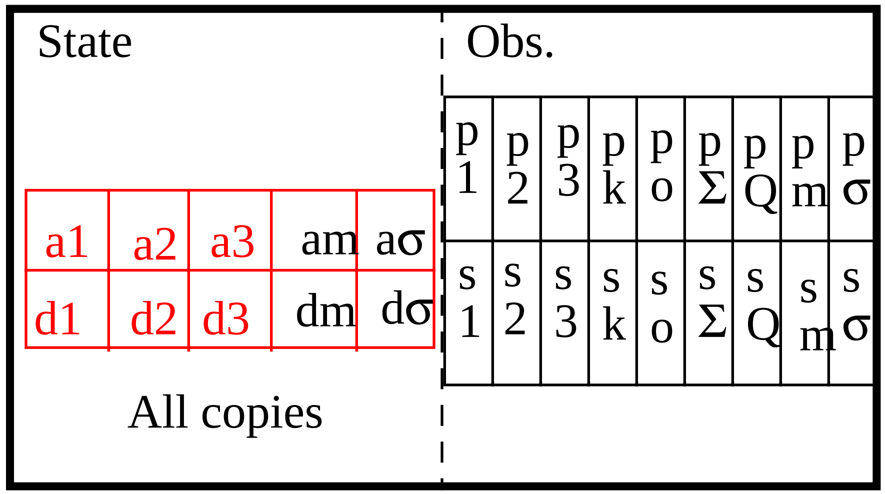
It computes observation space increments for p.

Broadcasts the prior p ensemble and the increments.



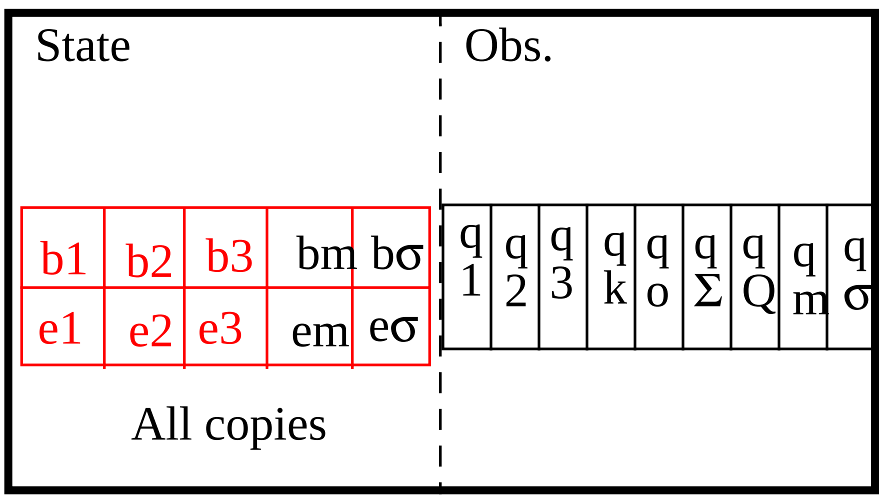
All PEs now have prior and increment for observation p.

PE0

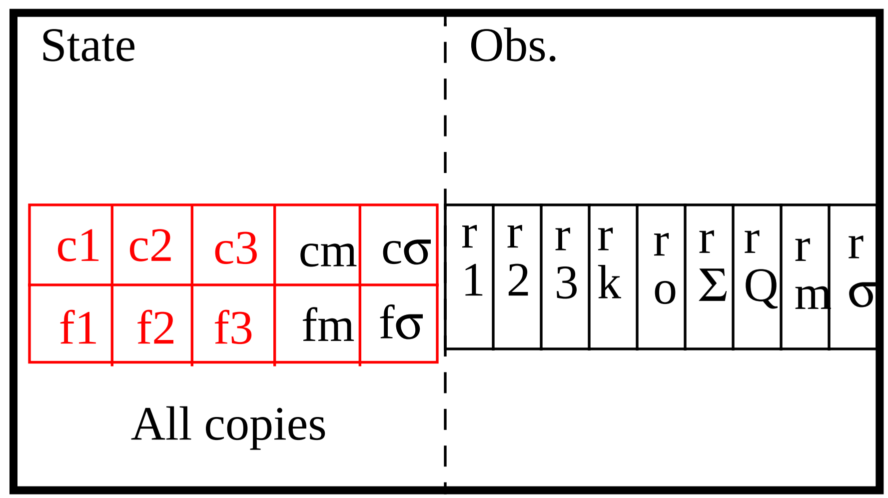


Each PE (including PE0) finds all of its state variables that are close to p.
For instance, PE1 checks to see if b or e are close to p.
PE1 then uses regression to update ensemble members for b and e as needed.

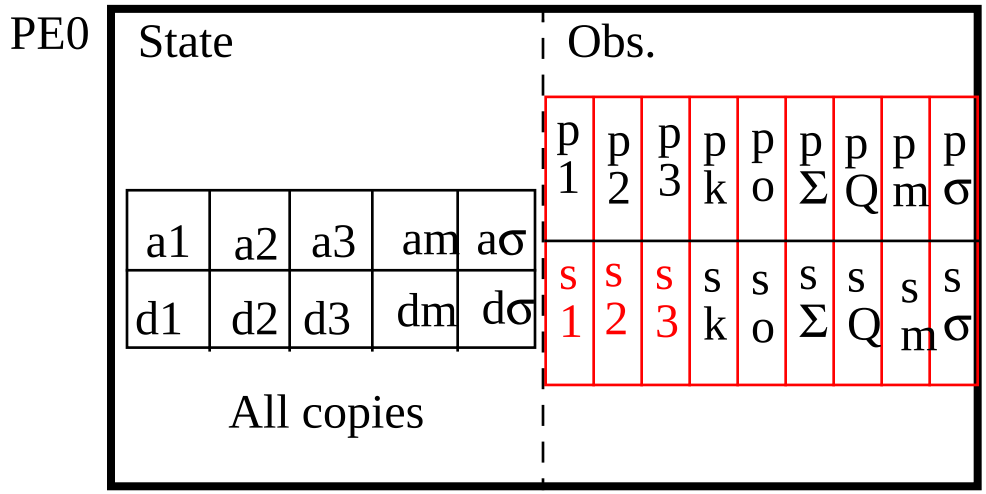
PE1



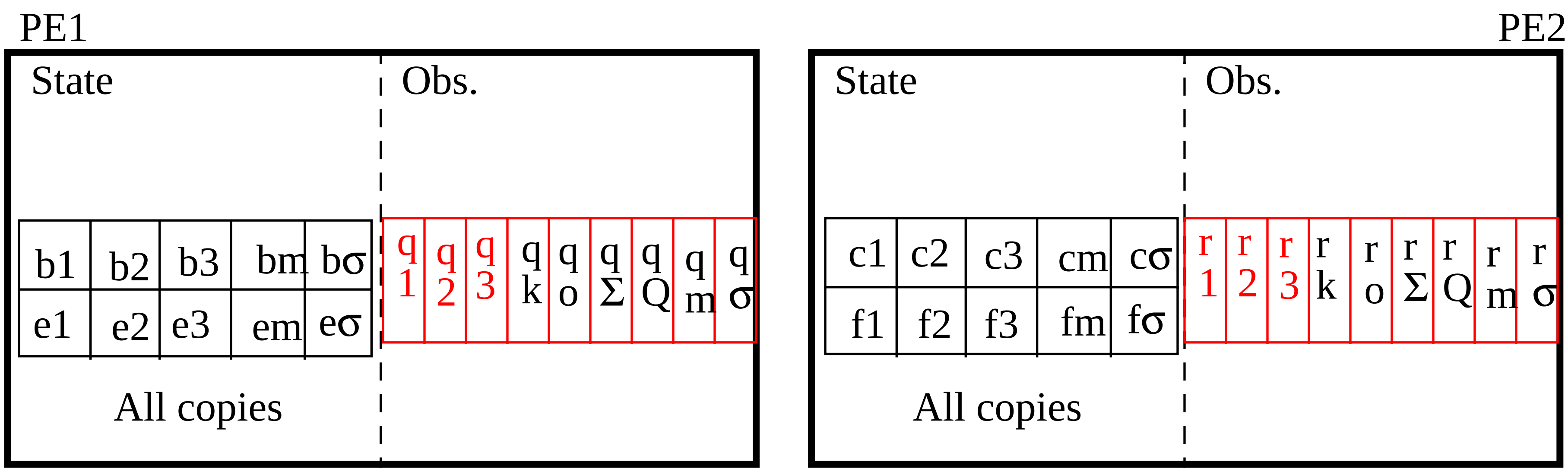
PE2



All PEs now have prior and increment for observation p.



Each PE next updates the prior for each of its observations that are close to p.
No need to do this for observations that have already been used (like p here).
For instance, PE1 updates q by regression if it is close to p.

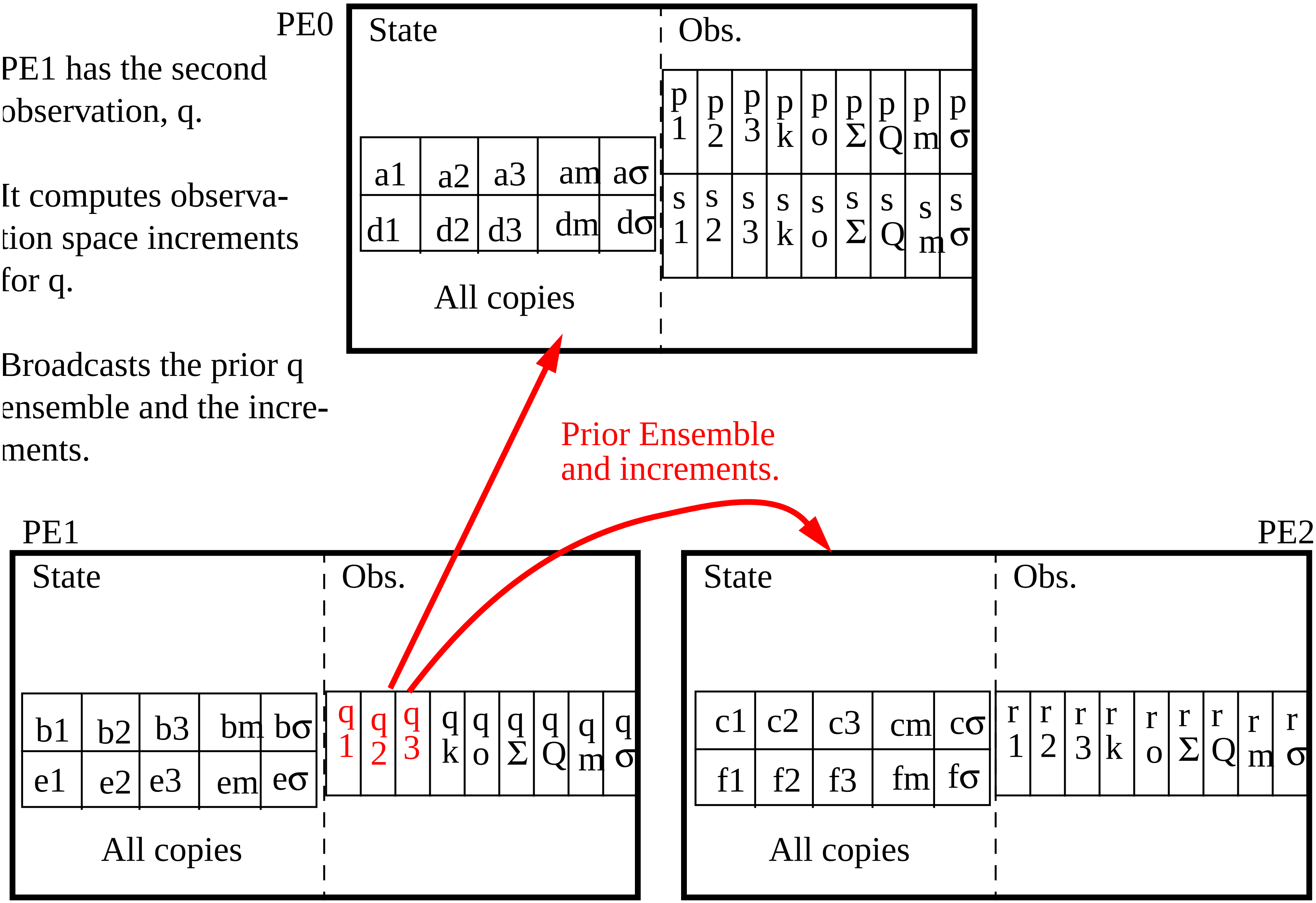


When everyone is finished regressing for observation p, assimilate q.

PE1 has the second observation, q.

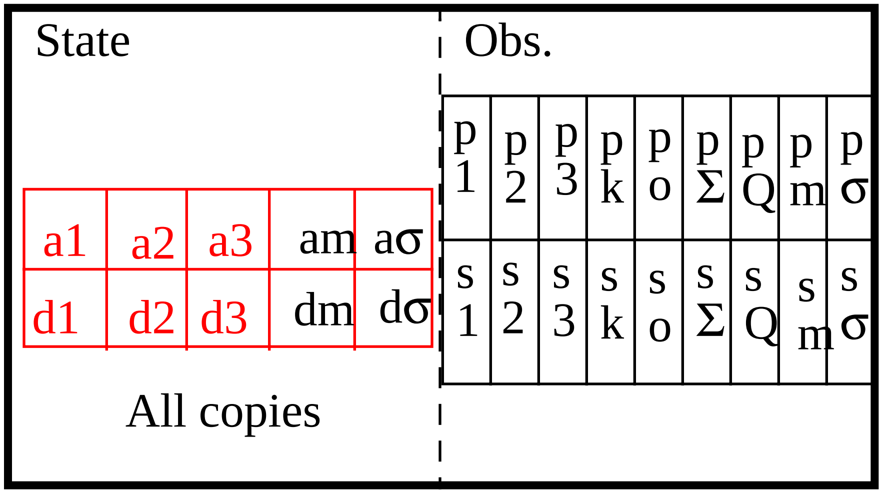
It computes observation space increments for q.

Broadcasts the prior q ensemble and the increments.



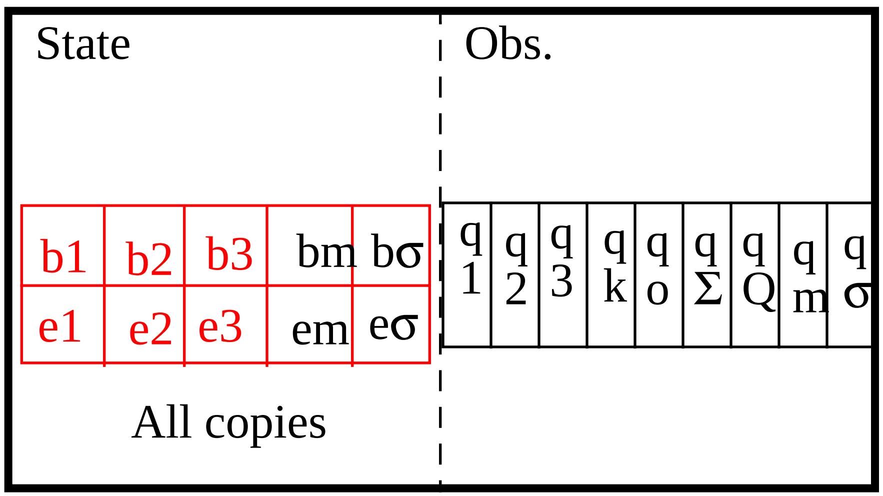
All PEs now have prior and increment for observation q.

PE0

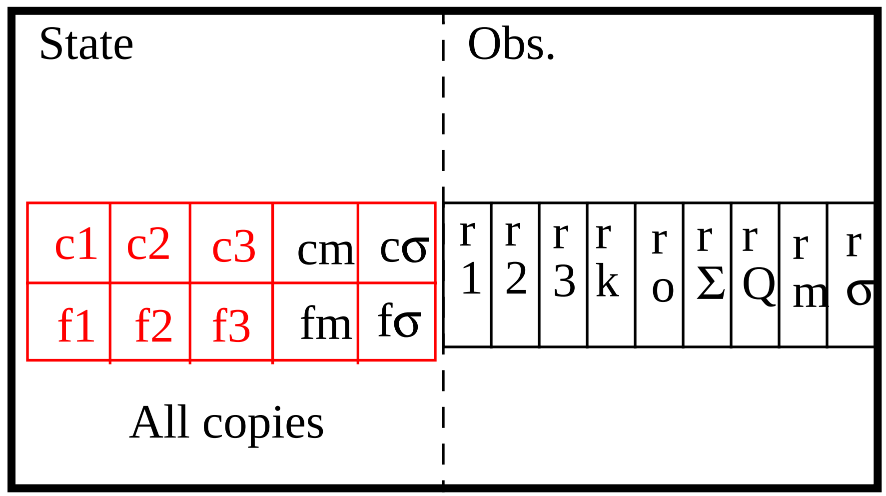


Each PE (including PE1) finds all of its state variables that are close to q.
For instance, PE0 checks to see if a or d is close to q.
PE0 then uses regression to update ensemble members for a and d as needed.

PE1

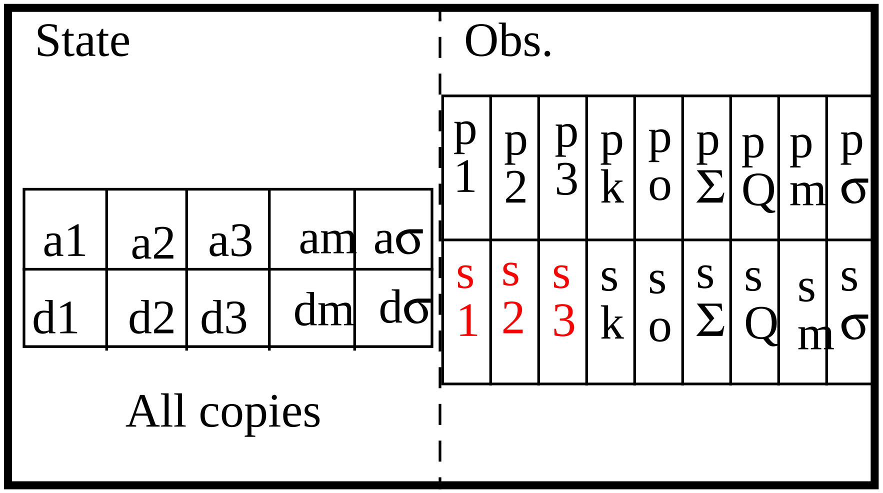


PE2



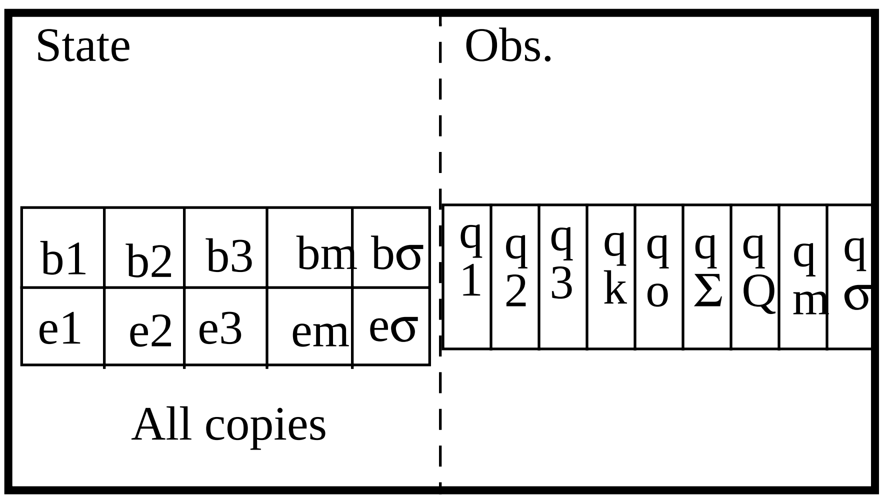
All PEs now have prior and increment for observation q.

PE0

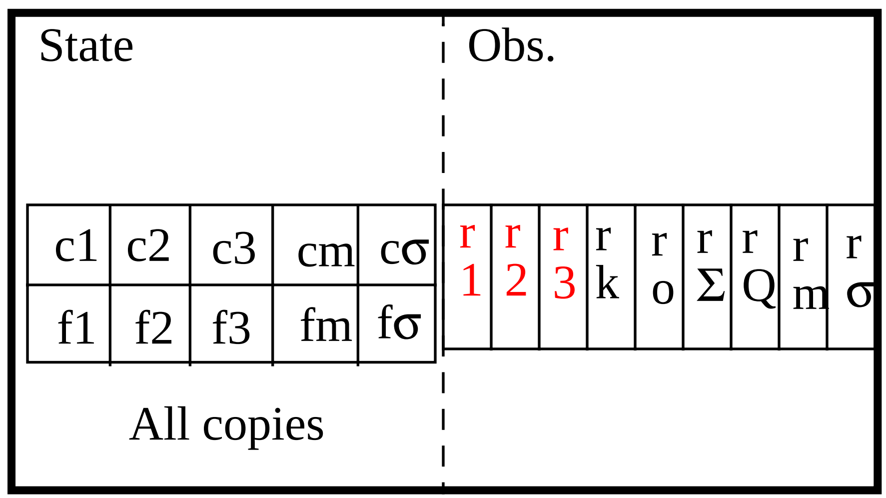


Each PE next updates the prior for each of its observations that are close to q.
No need to do this for observations already used (like p and q here).
For instance, PE0 updates s by regression if it is close to q.
Continue until all observations have been used (r and s remain here).

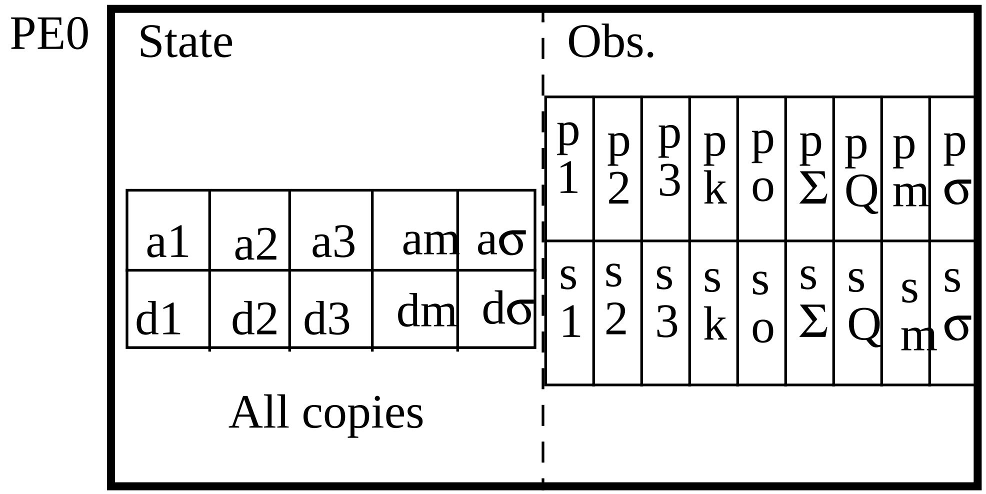
PE1



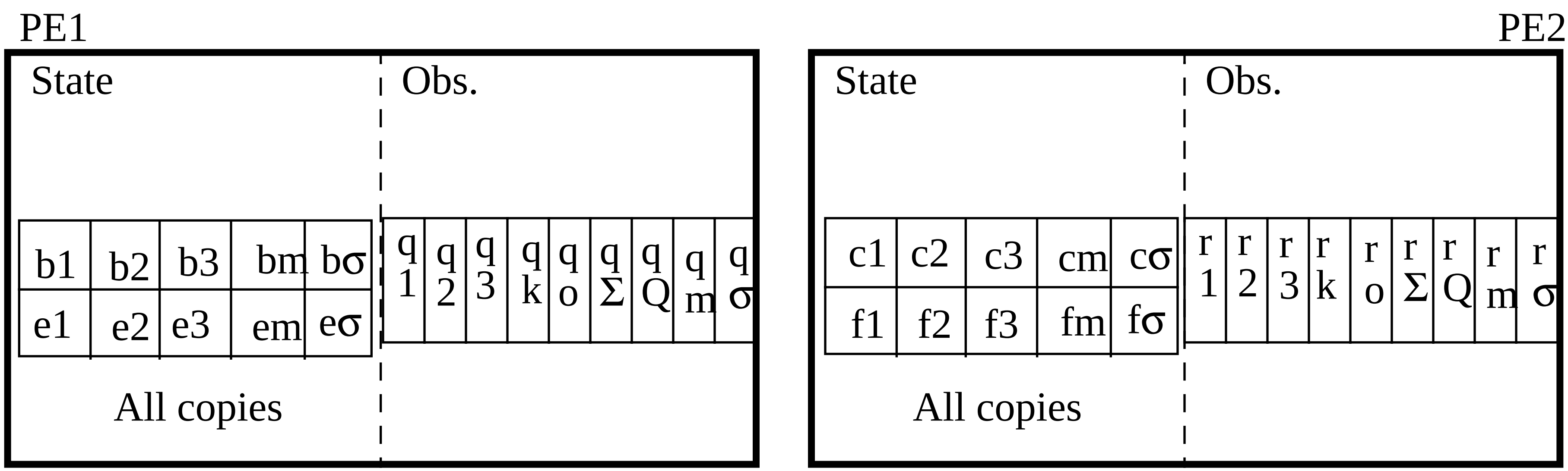
PE2



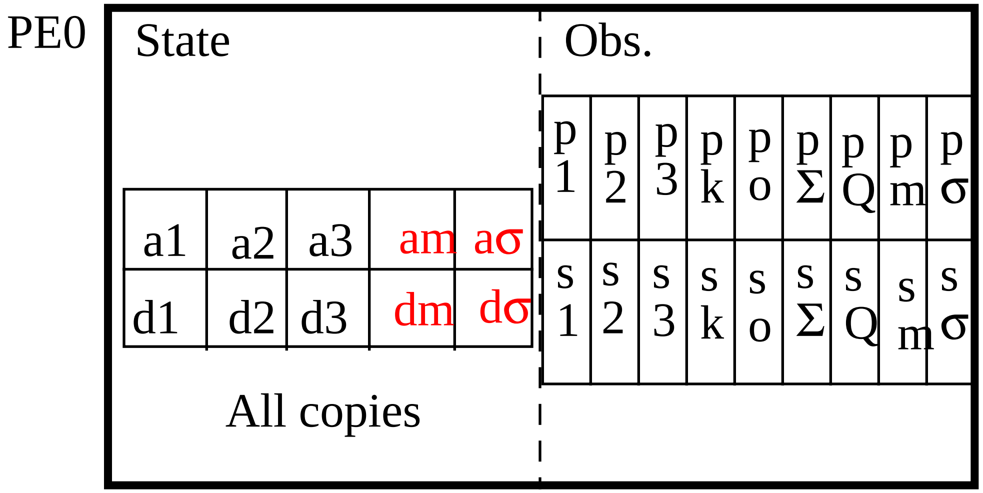
Locating close observations and states; computing distances.



Note that PEs do NOT have the complete state vector during assimilation.
Models that require state vector to compute locations have problems.
They can store ensemble mean that was computed and broadcast earlier.

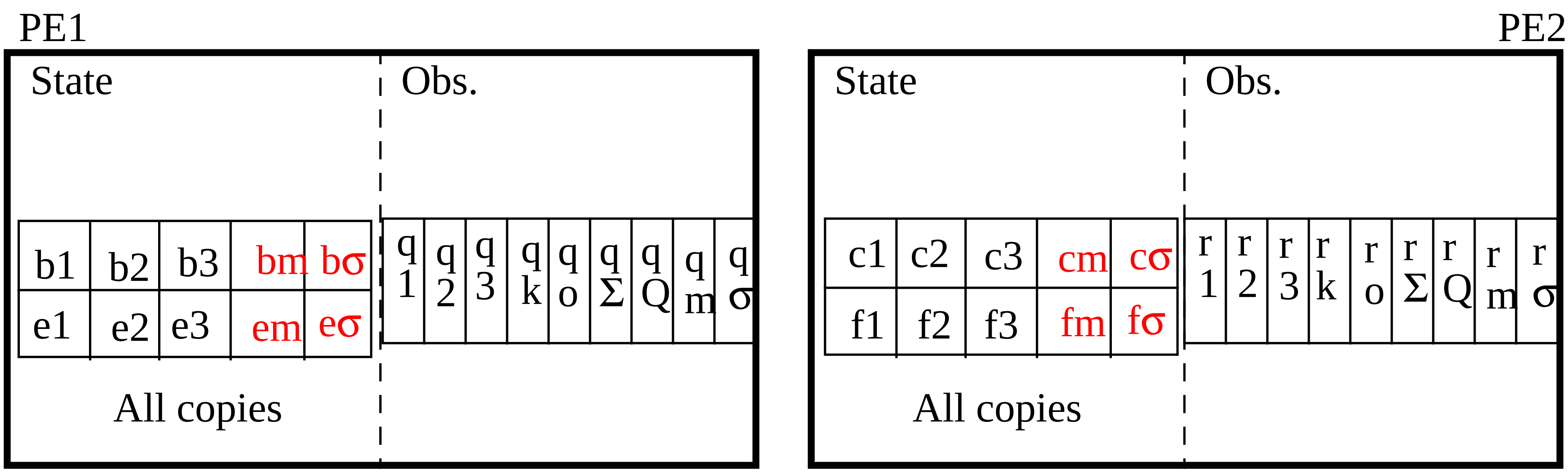


filter_assim is now complete. Control returns to filter_main in filter.f90.



Update the ensemble mean and s.d. for the state variables.

Each processor computes this for its subset of variables.



Completing steps are similar to those already discussed.

Transform state to have subset of copies of all state variables on each PE.

Do posterior state space diagnostics as per prior.

Compute forward observation operators as was done before assimilation.

Do observation space posterior diagnostics as per prior.

If there are more observation times, go back to model advance.

If no more observation times, output state restart files.