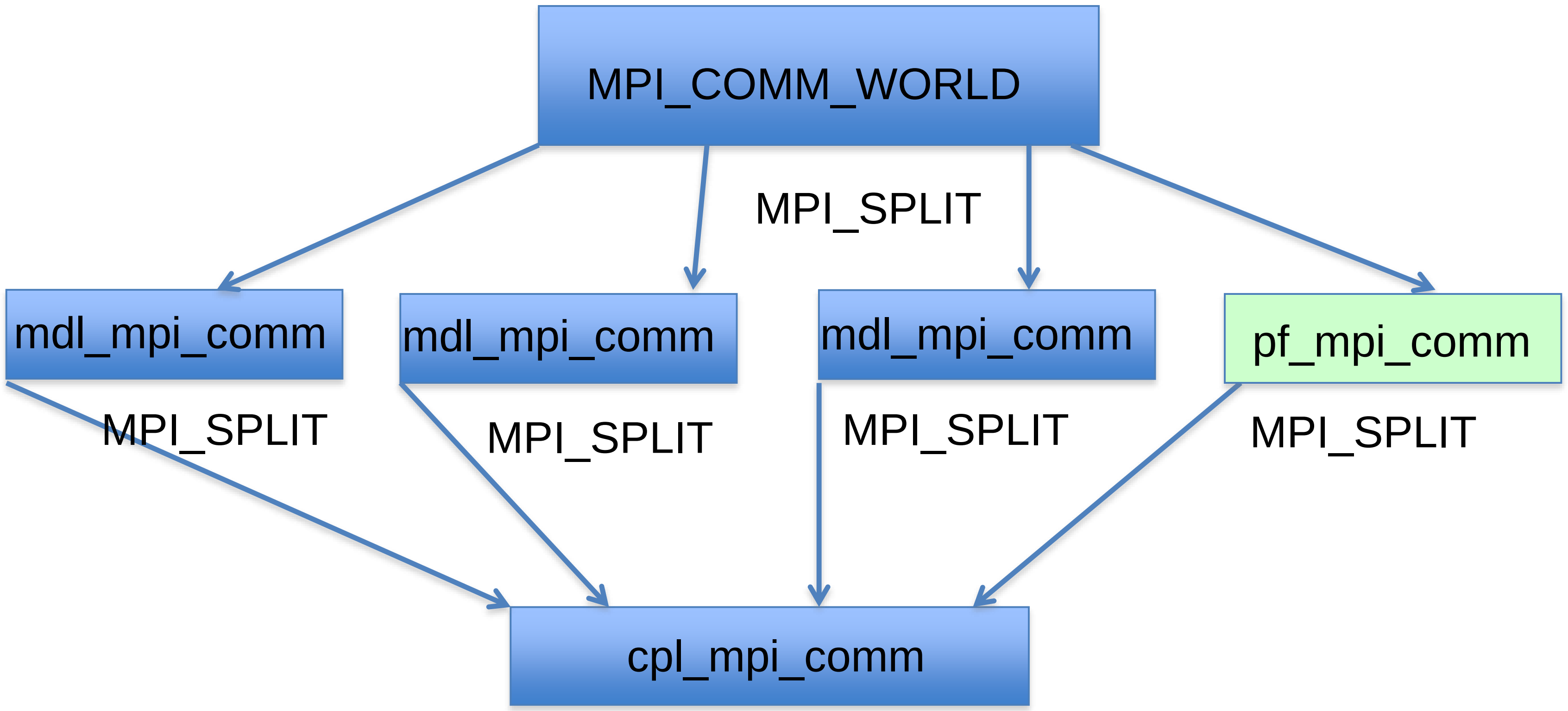
Ensemble data assimilation framework @ Reading: EMPIRE

Peter Jan van Leeuwen, Phil Browne,
Simon Wilson, Sanita Vetra-Carvalho
Data-Assimilation Research Centre DARC
University of Reading

Basic idea: communication via MPI



```
subroutine initialise_mpi (mdl_id , cpl_root , cpl_mpi_comm )
    implicit none
    include 'mpif .h’
    integer :: mdl_num_proc =1
    integer :: mpi_err ,cpl_root , CPL_MPI_COMM ,tag
    integer :: mpi_status ( MPI_STATUS_SIZE )
    integer :: da ,nda ,ptcl_id ,nens ,gbl_id
    integer :: couple_colour ,mdl_id , mdl_mpi_comm ,mdlcolour

    call mpi_init ( mpi_err )
    CALL MPI_COMM_RANK ( MPI_COMM_WORLD ,gbl_id ,mpi_err )
    mdlcolour = gbl_id / mdl_num_proc
    call mpi_comm_split(MPI_COMM_WORLD,mdlcolour,gbl_id,mdl_mpi_comm,
                                                            mpi_err )
    call mpi_comm_rank ( mdl_mpi_comm ,mdl_id ,mpi_err )
    if( mdl_id .eq. 0) then
        couple_colour = 9999
    else
       couple_colour = mpi_undefined
    end if
```

```
 CALL MPI_COMM_SPLIT (MPI_COMM_WORLD,couple_colour ,gbl_id ,CPL_MPI_COMM,
                                                                mpi_err )
    if( mdl_id .eq. 0) then
        call mpi_comm_size ( cpl_mpi_comm ,nens ,mpi_err )
        call mpi_comm_rank ( cpl_mpi_comm ,ptcl_id ,mpi_err )
        da = 0
        call mpi_allreduce (da ,nda ,1, mpi_integer ,mpi_sum ,
                   cpl_mpi_comm ,mpi_err )
        nens = nens - nda
        do da = 1,nda
           if( real ( ptcl_id ) .lt. real (nens * (da))/ real (nda )) then
               cpl_root = da -1+ nens
              exit
           end if
        end do
    else
        cpl_root = -1
    end if
end subroutine initialise_mpi
```

```
program  lorenz63
implicit none
include  'mpif .h’
real  ( kind  = kind  (1.0 D0 )) :: t,sigma ,rho ,beta ,dt ,tstart , tstop
real  ( kind  = kind  (1.0 D0 )), dimension  (3) :: x,k1 ,k2 ,k3 ,k4
integer  :: mpi_err ,mdl_id , cpl_root , mpi_status , cpl_mpi_comm

call initialise_mpi (mdl_id , cpl_root , cpl_mpi_comm )

tstart =0.0 D0 ; dt = 0.01 D0 ; tstop = real  (40*100)* dt

sigma = 10.0 D0 ; rho = 28.0 D0 ; beta = 8.0 D0 /3.0 D0

x = (/ 1.508870 D0 , -1.531271 D0 , 25.46091 D0 /)

call mpi_send(x,3,mpi_double_precision,cpl_root,1,cpl_mpi_comm,mpi_err)
call mpi_recv(x,3,mpi_double_precision,cpl_root,1,cpl_mpi_comm,mpi_status,
mpi_err)
```

```
t = tstart
do; if ( t .ge. tstop -1.0D -10) exit
   k1 = f (x , sigma , rho , beta )
   k2 = f (x +0.5 D0 * dt * k1 , sigma , rho , beta )
   k3 = f (x +0.5 D0 * dt * k2 , sigma , rho , beta )
   k4 = f (x + dt * k3 , sigma , rho , beta )
   x = x + dt *( k1 + 2.0 D0 *( k2 + k3 ) + k4 )/6.0 D0
   call mpi_send(x,3,mpi_double_precision,cpl_root,1,cpl_mpi_comm,mpi_err)
   call mpi_recv(x,3,mpi_double_precision,cpl_root,1,cpl_mpi_comm,
                                                       mpi_status,mpi_err )
   t = t + dt
   print *,x(1) ,x(2) ,x(3)
end do
call mpi_finalize ( mpi_err )
contains
function  f (x , sigma , rho , beta )
   implicit none
   real  ( kind  = kind  (1.0 D0 )), intent  (in ), dimension  (3) :: x
   real  ( kind  = kind  (1.0 D0 )), dimension  (3) :: f
   real  ( kind  = kind  (1.0 D0 )), intent  (in ) :: sigma , rho , beta
    f = (/ sigma*(x(2)-x(1)),x(1)*(rho -x (3))-x(2),x(1)*x(2)-beta*x (3)/)
end function  f
subroutine initialise_mpi
end program lorenz63
```

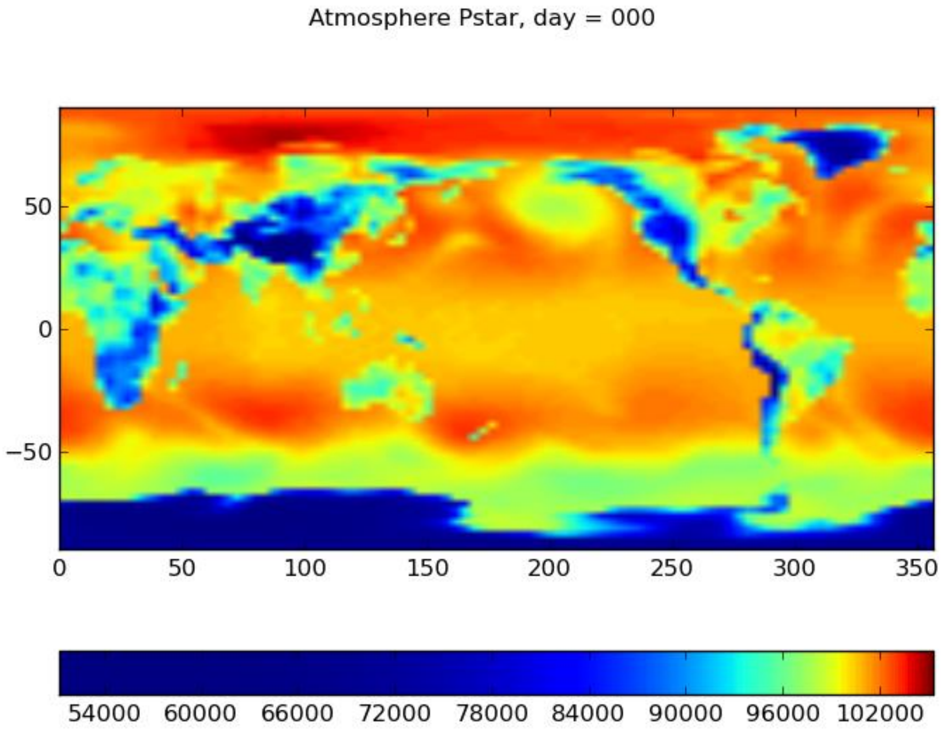
Implementation examples

- Barotropic vorticity equation 65,536 variables
- TELEMAC unstructured finite element for North Sea, 114,288 variables
- HadCM3 coupled atmosphere-ocean climate model, 2,314,430 variables
- Unified Model atmospheric NWP model, 275,537,920 variables

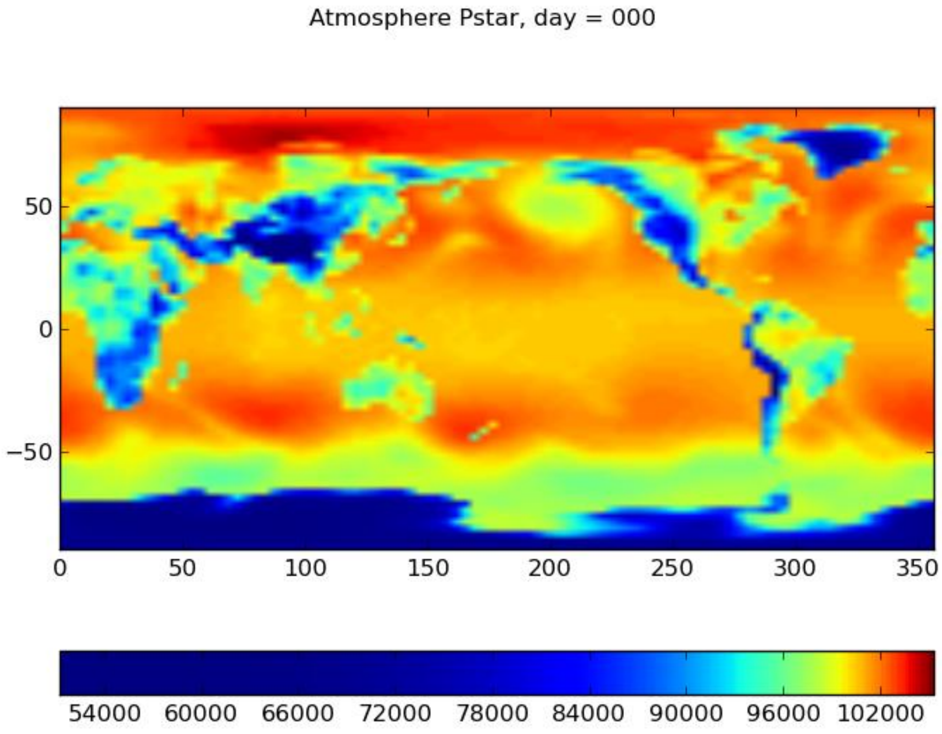
Timings for HadCM3

Number of ensemble members	Number of cores used	No communication time (s)	Communication only time (s)	Equal weight filter time (s)
1	64	13.1	14.5	159.8
3	192	13.1	15.0	164.4
7	448	13.1	15.5	165.2
15	960	13.1	16.0	163.8
31	1984	13.1	18.0	167.1
62	3968	13.1	21.5	174.2

Atmospheric surface pressure

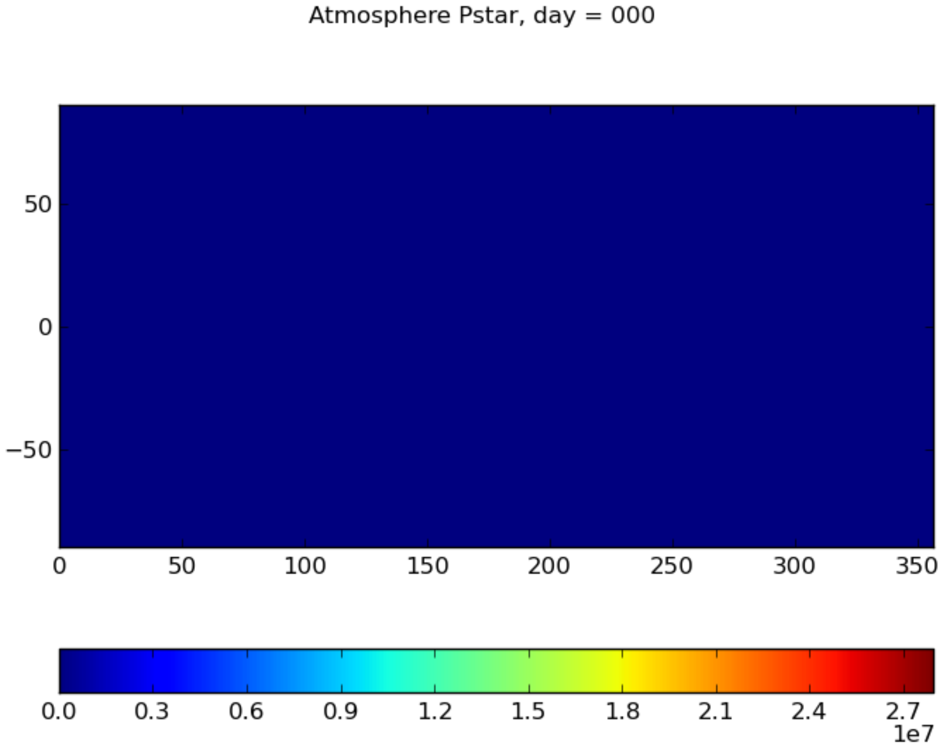


Truth

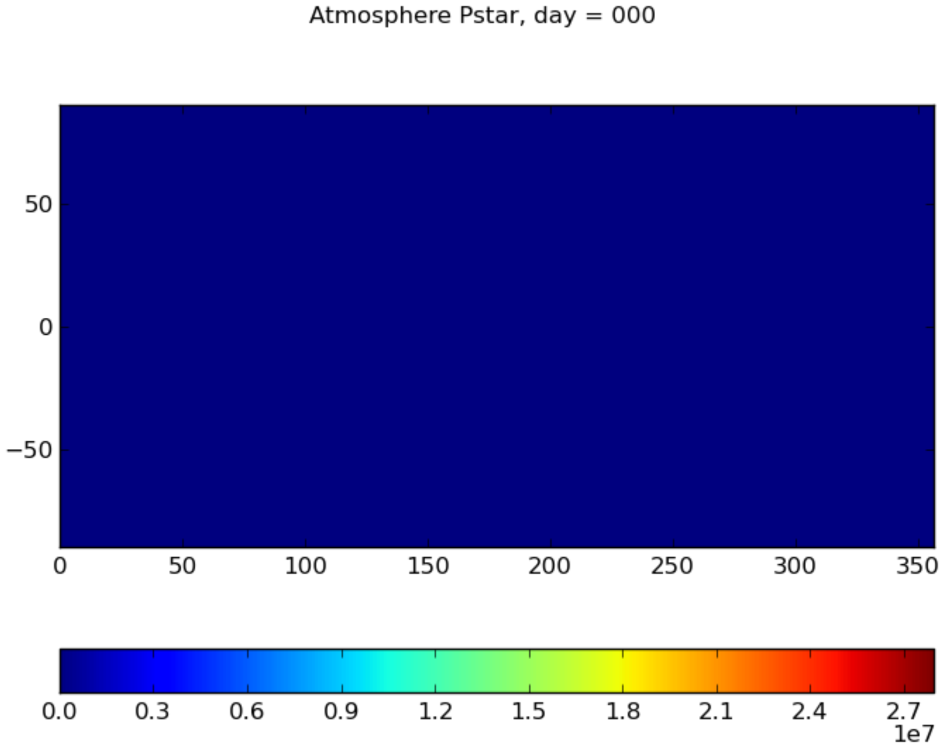


Particle filter mean

Atmospheric Surface pressure

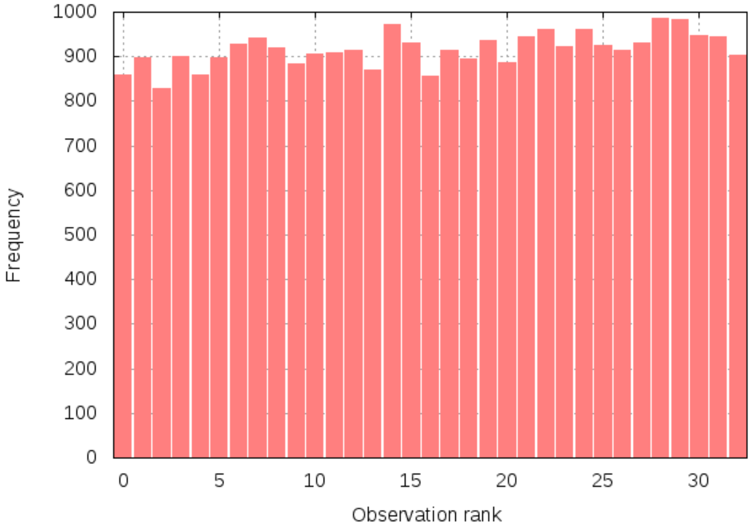


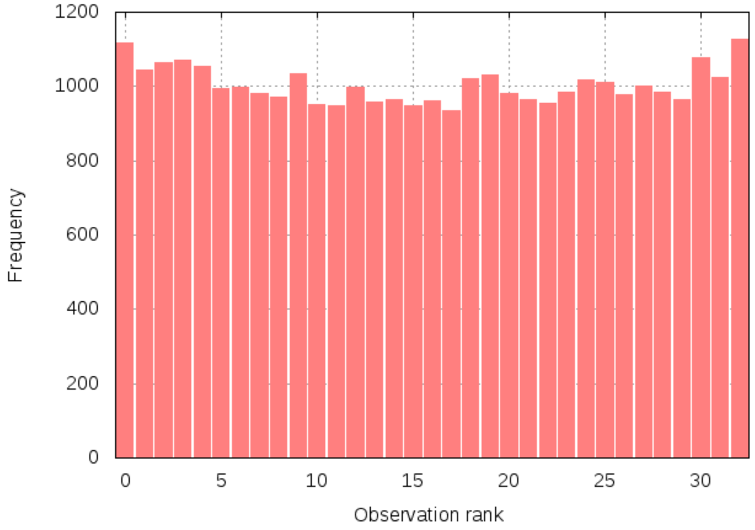
Particle filter variance

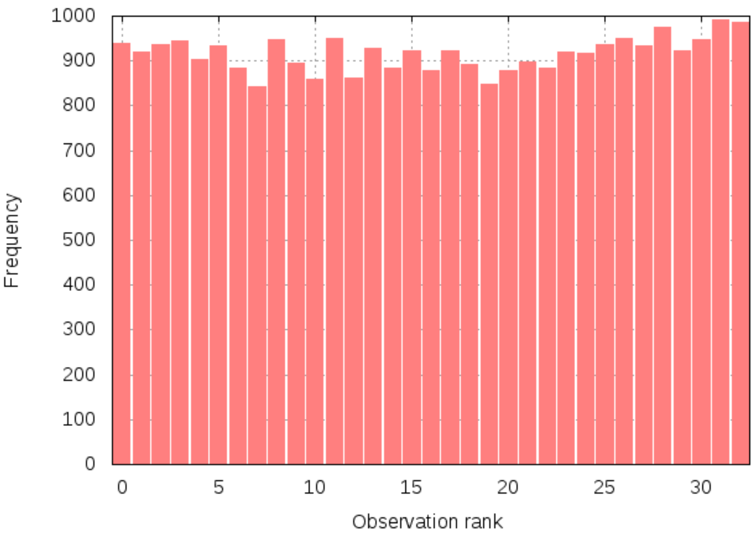


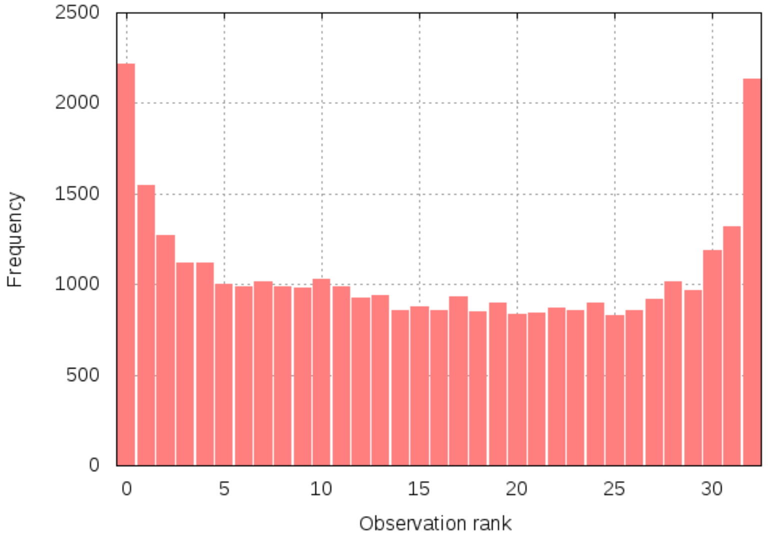
(Truth-mean)^2

Rank histograms

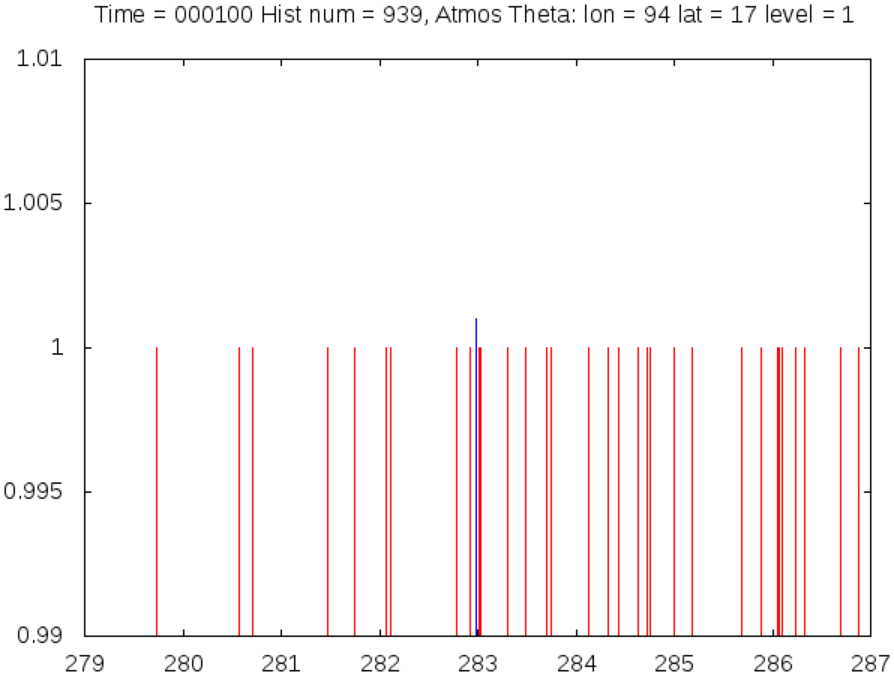

Histogram 1 - Atmosphere P* surface


Histogram 37 - Ocean V level 1 surface

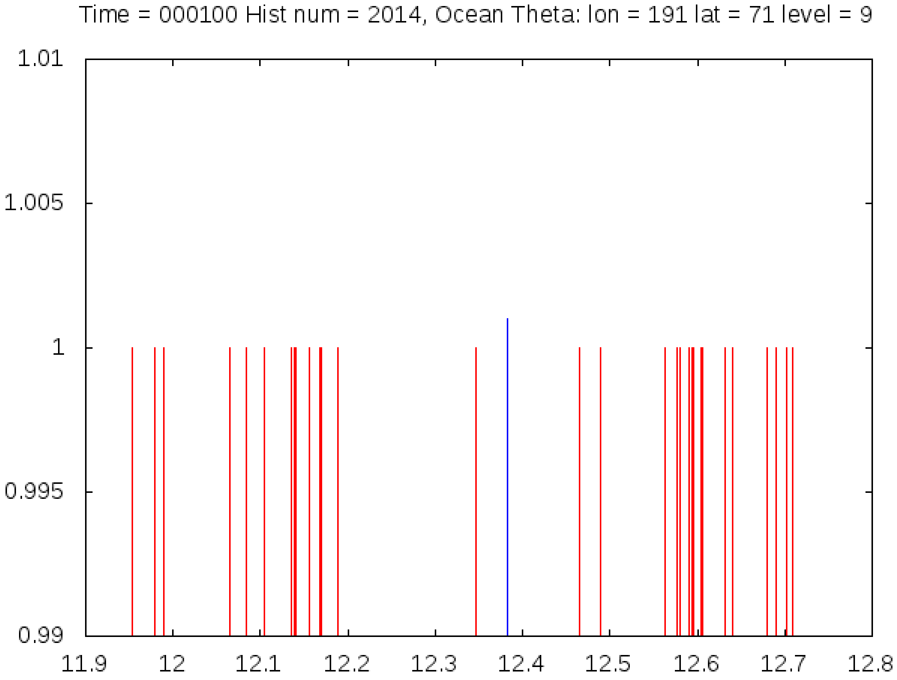

Histogram 7 - Atmosphere V level 1 surface


Histogram 22 - Ocean Theta level 1 surface

Pdf’s



Atmosphere point pdf day 100



Ocean point pdf day 100

Conclusions

- EMPIRE is a efficient ensemble data-assimilation framework
- Proposal density in particle filtering allows enormous freedom
- Equivalent-weights scheme ‘solves’ dimensionality problem
- but needs tuning…
- Other efficient schemes are being derived.
- Present work: numerical weather prediction, climate forecasting