CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Summer School 2015

MPI Profiling
http://github.com/eth-cscs/SummerSchool2015/wiki - DAY4
July 2015

Summary

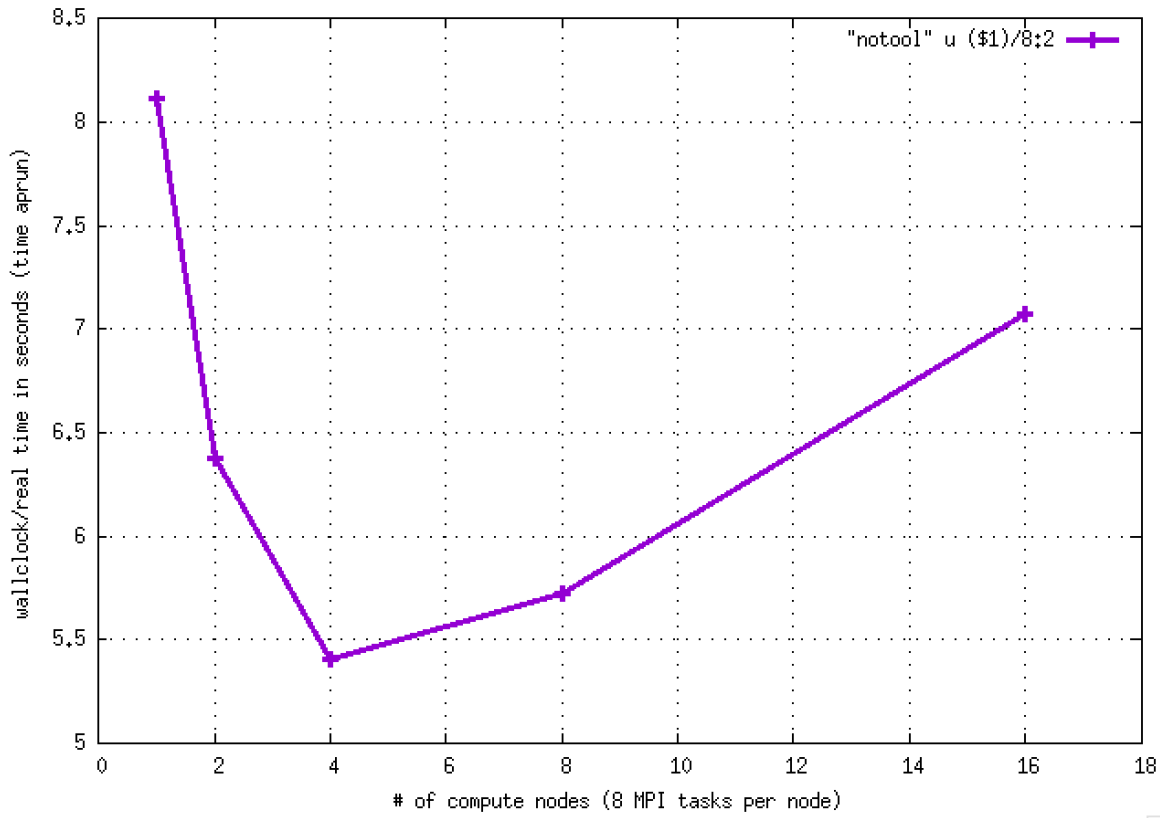
- Parallel performance
- Profiling the MPI mini app
  - with Cray's perftools-lite
  - with scorep

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Parallel Performance

Scalability results (C++/MPI miniapp)



Getting timings

```
course51@daint103:~ /usr/bin/time -p aprun -n 32 256 256 400 0.1
==========================================================
                      Welcome to mini-stencil!
version   :: with MPI : 32 MPI ranks
mesh      :: 256 * 256 dx = 0.00392157
time      :: 400 time steps from 0 .. 0.1
...  ----------------------------------------- Goodbye!
     real 5.41          <================ real time in seconds
```

WHY ?

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Perftools

MPI miniapp + perftools: compiling

Compiling with the tool

- Cray's perftools supports all 4 compilers
- module load `perftools-lite`; man `perftools-lite`
- make clean; make

Recompile with optimisation flags on

```
course51@daint103:~ make
CC -O3  -c stats.cpp -o stats.o
CC -O3  -c data.cpp -o data.o
CC -O3  -c operators.cpp -o operators.o
CC -O3  -c linalg.cpp -o linalg.o
CC -O3 *.o main.cpp -o main.exe

INFO: creating the CrayPat-instrumented exec 'main.exe'
                          (sample_profile) ...OK

INFO: A maximum of 53 functions from group 'io' will be traced.
INFO: A maximum of 292 functions from group 'mpi' will be traced.
INFO: A maximum of 22 func. from group 'realtime' will be traced.
INFO: A maximum of 54 func. from group 'syscall' will be traced.
```

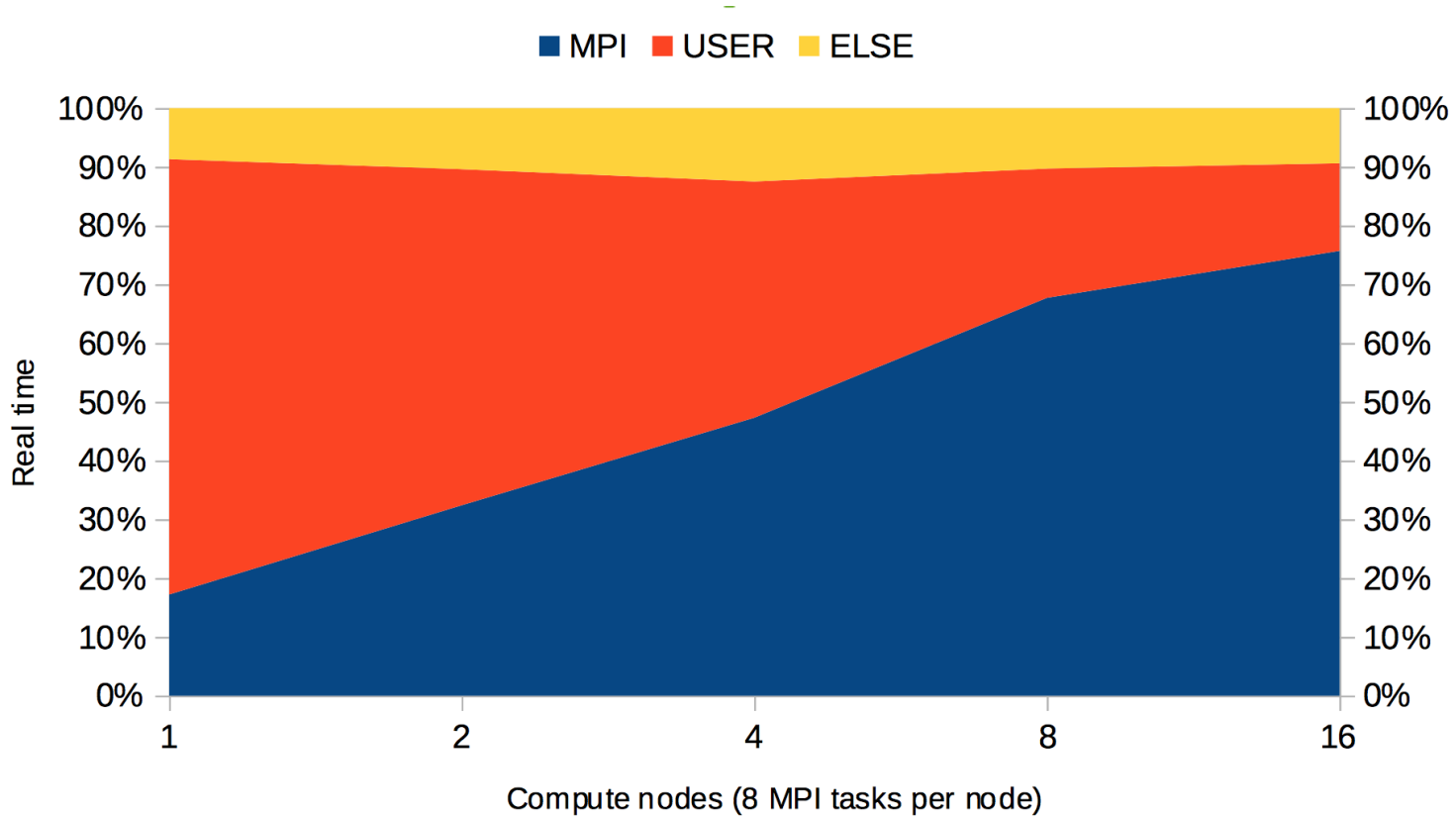
MPI miniapp + perftools: running

Run to get the performance report (4 CN here)

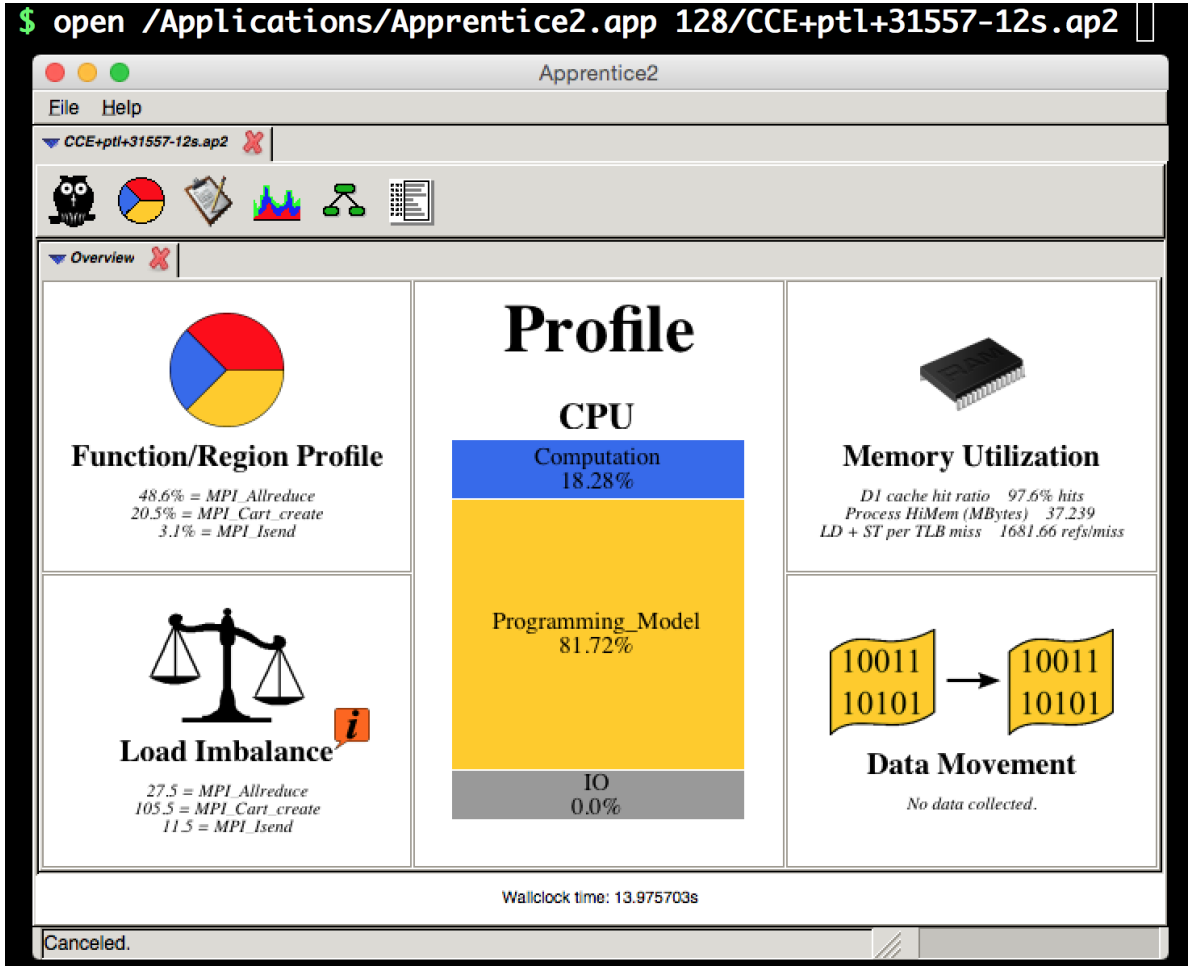
```
course51@daint103:~ aprun -n 32 256 256 400 0.1

=================================================
                    Welcome to mini-stencil!
...
------------------------------------------------- Goodbye!
Table 1:  Profile by Function Group and Function
          (top 10 functions shown)
  Samp% |   Samp | Imb. |  Imb. |Group
        |        | Samp | Samp% | Function
        |        |      |       |  PE=HIDE

 100.0% | 444.3 |   -- |    -- |Total
|---------------------------------------
|  47.3% | 210.2 |   -- |    -- |MPI
||--------------------------------------
||  34.2% | 151.9 | 29.1 | 16.6% |MPI_Allreduce
||   6.1% |  27.1 | 17.9 | 41.1% |MPI_Cart_create
||   3.2% |  14.4 | 14.6 | 51.8% |MPI_Isend
||   3.0% |  13.4 | 10.6 | 45.4% |MPI_Waitall
||======================================
|  40.2% | 178.8 |   -- |    -- |USER
|   7.7% |  34.1 |   -- |    -- |OMP
|   4.8% |  21.2 |   -- |    -- |ETC
```

MPI miniapp + perftools: Because !

Scalability results (C++/MPI miniapp): 256x256



MPI miniapp + perftools: apprentice2

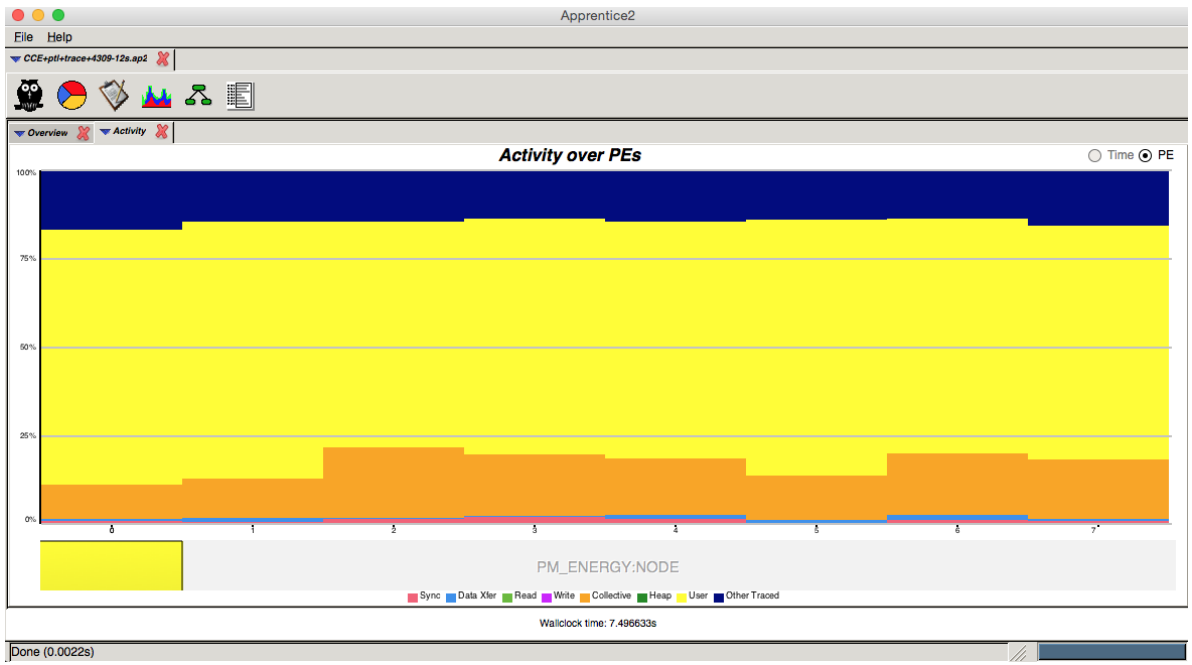


- You can install app2 on your laptop: `/opt/cray/perftools/default/share/desktop_installers/`

MPI miniapp + perftools: tracing

app2

- module load `perftools-lite`; man `perftools-lite`
- export CRAYPAT_LITE=event_profile
- make clean; make; aprun ...



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Scorep

MPI miniapp + scorep: compiling

Compiling with the tool

- module rm perftools-lite
- module load scorep; module help scorep
- make clean; make CXX=’`scorep --mpp=mpi` CC’

Recompile with optimisation flags on

```
course51@daint103:~ make CXX="scorep --mpp=mpi CC" CXXFLAGS="-O3 -h
      noomp"
         scorep --mpp=mpi CC -O3 -h noomp  -c stats.cpp
         scorep --mpp=mpi CC -O3 -h noomp  -c data.cpp
         scorep --mpp=mpi CC -O3 -h noomp  -c operators.cpp
         scorep --mpp=mpi CC -O3 -h noomp  -c linalg.cpp
         scorep --mpp=mpi CC -O3 -h noomp *.o main.cpp -o main
```

MPI miniapp + scorep: running

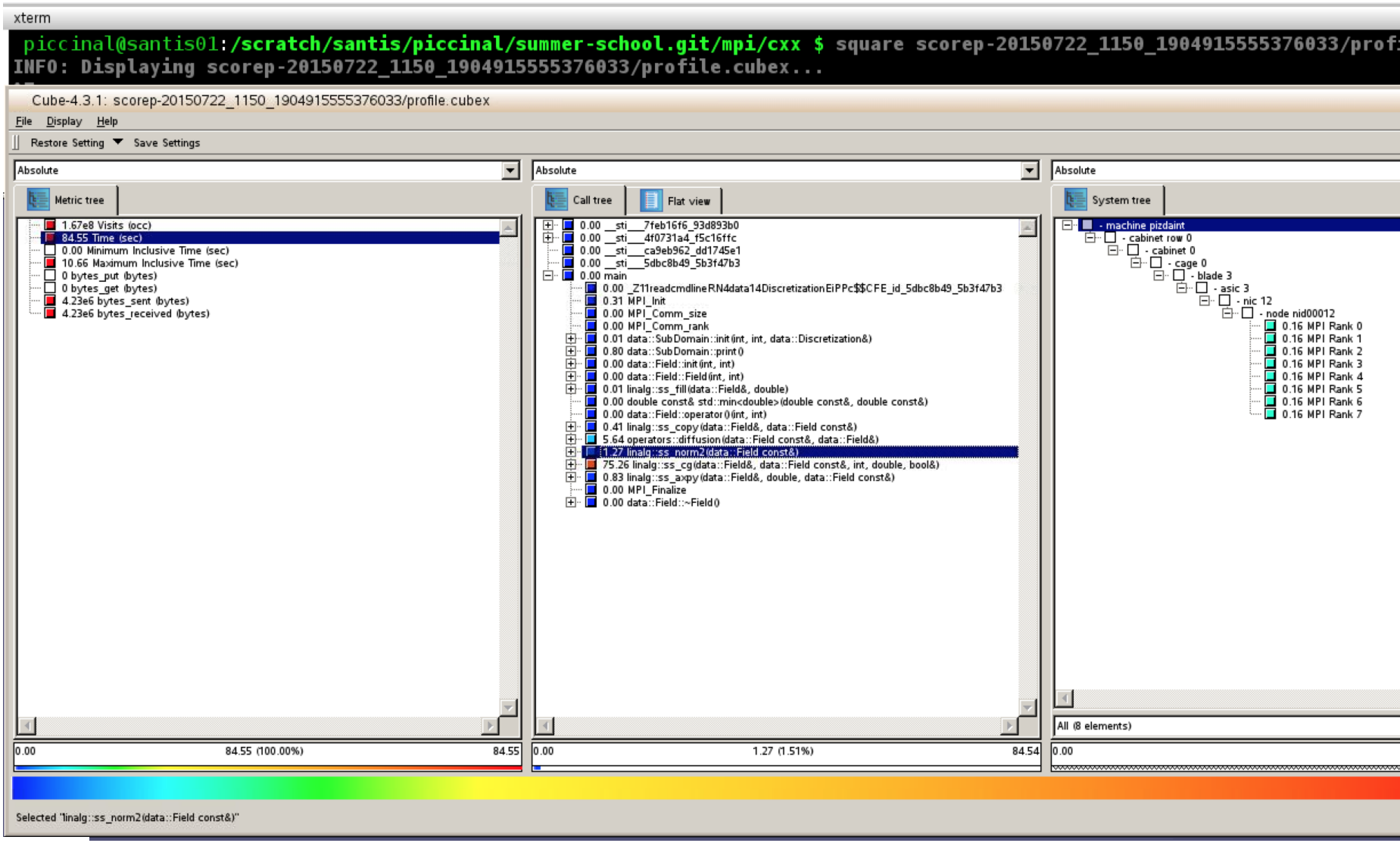
Run to get the performance report dir

```
course51@daint103:~ aprun -n 8 CCE+sc141 128 128 25 0.0001

=================================================
Welcome to mini-stencil!
...
------------------------------------------------- Goodbye!

New dir ==> scorep-20150722_1150_1904915555376033/
```

View the text performance report

```
course51@daint103:~ scorep-score \
   scorep-20150722_1150_1904915555376033/profile.cubex

 type   max_buf[B]       visits time[s] time[%] time/visit[us]
      region
  ALL 501,600,399 166,898,392   84.55   100.0           0.51   ALL
  USR 501,322,776 166,859,984   60.97    72.1           0.37   USR
  MPI     247,839      28,480    0.72     0.9          25.31   MPI
  COM      29,784       9,928   22.85    27.0        2301.87   COM
```

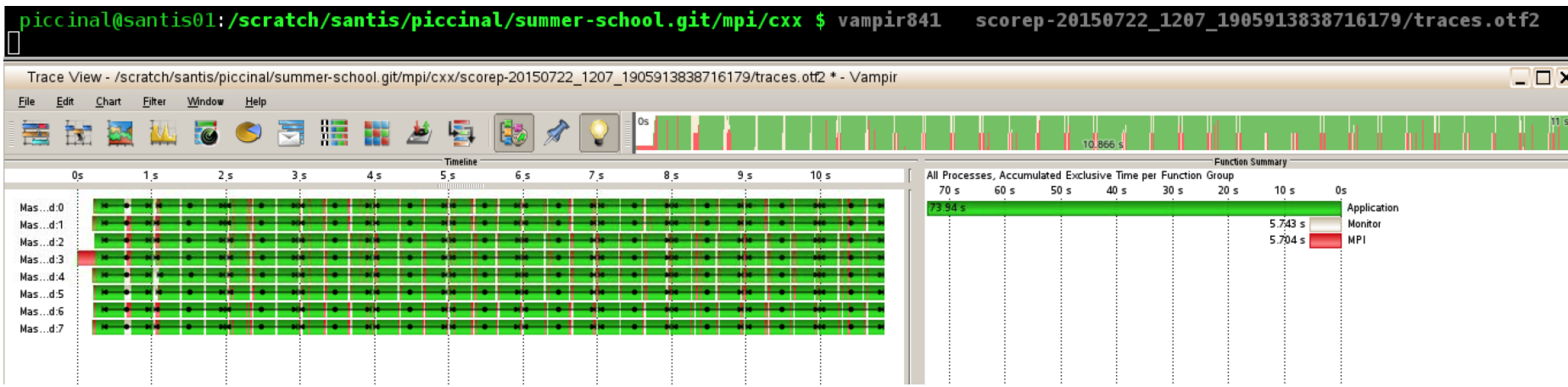
MPI miniapp + scorep: scalasca



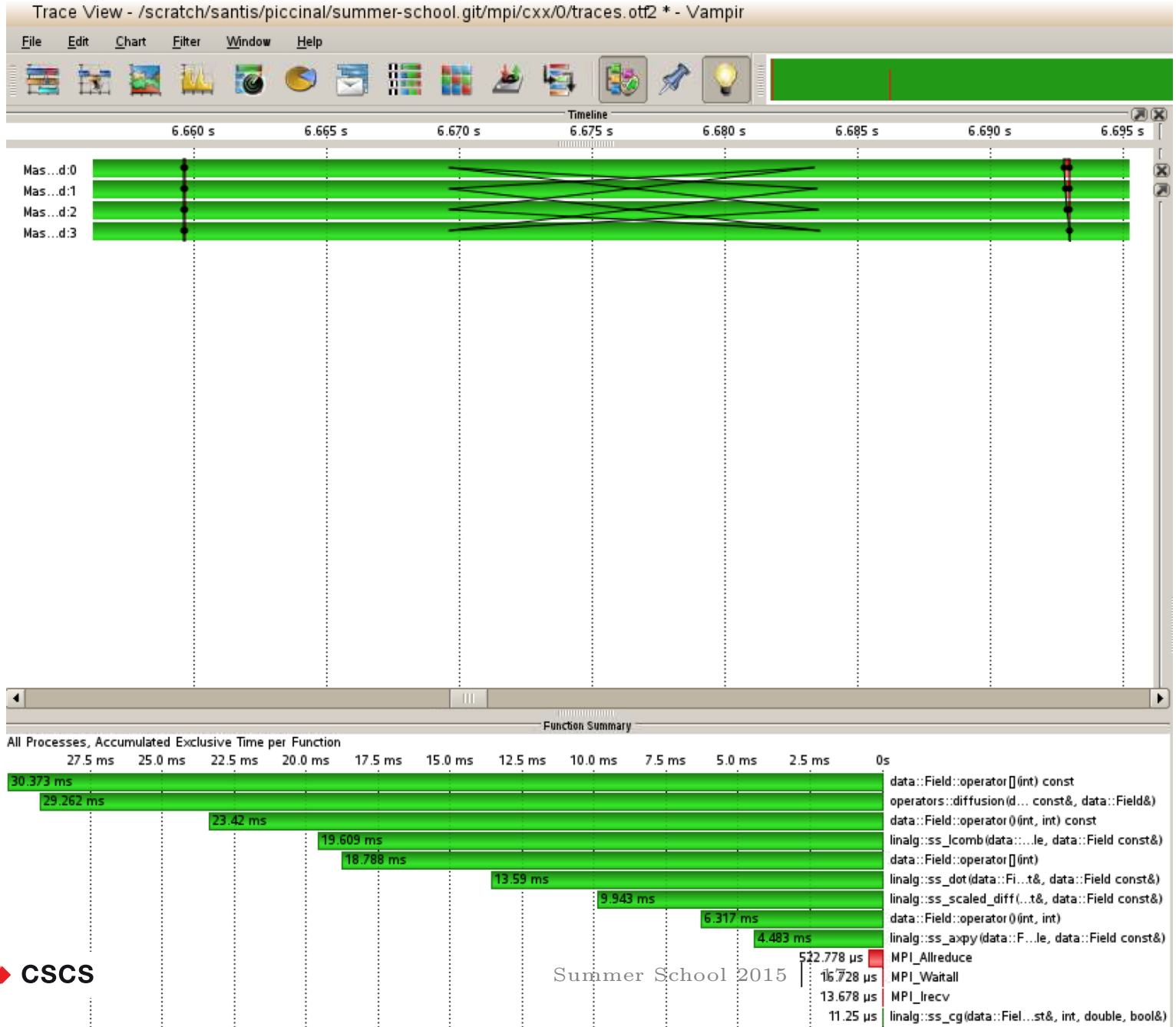
MPI miniapp + scorep: vampir (1)

vampir

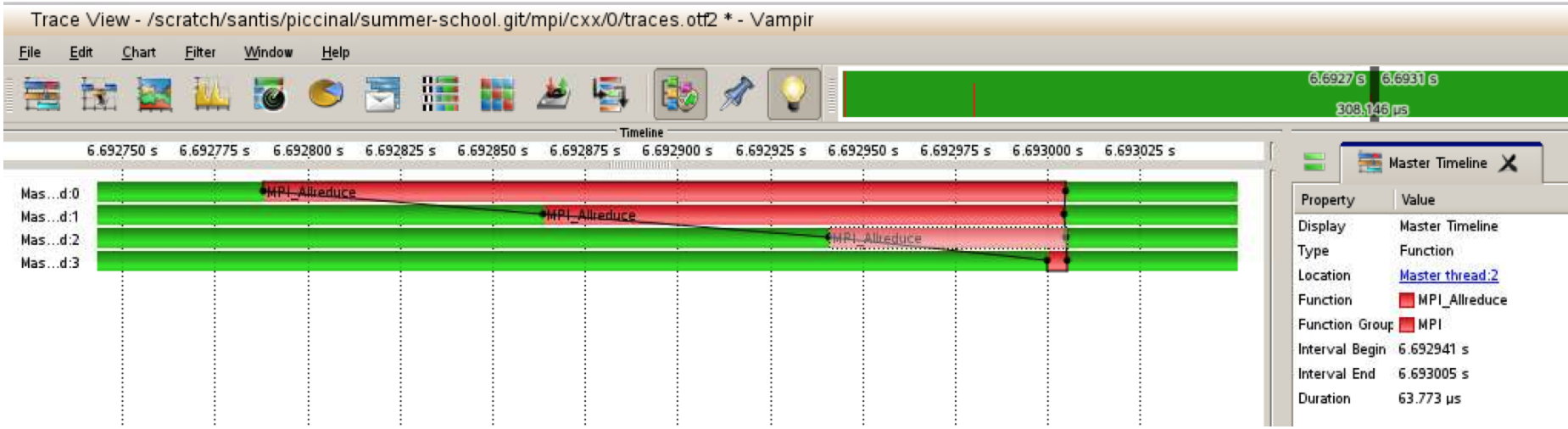
- module rm perftools-lite
- module load scorep; module help scorep
- make clean; make;
- export SCOREP_ENABLE_TRACING=true; aprun ...



MPI miniapp + perftools: vampir (2)



MPI miniapp + perftools: vampir (3)



CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

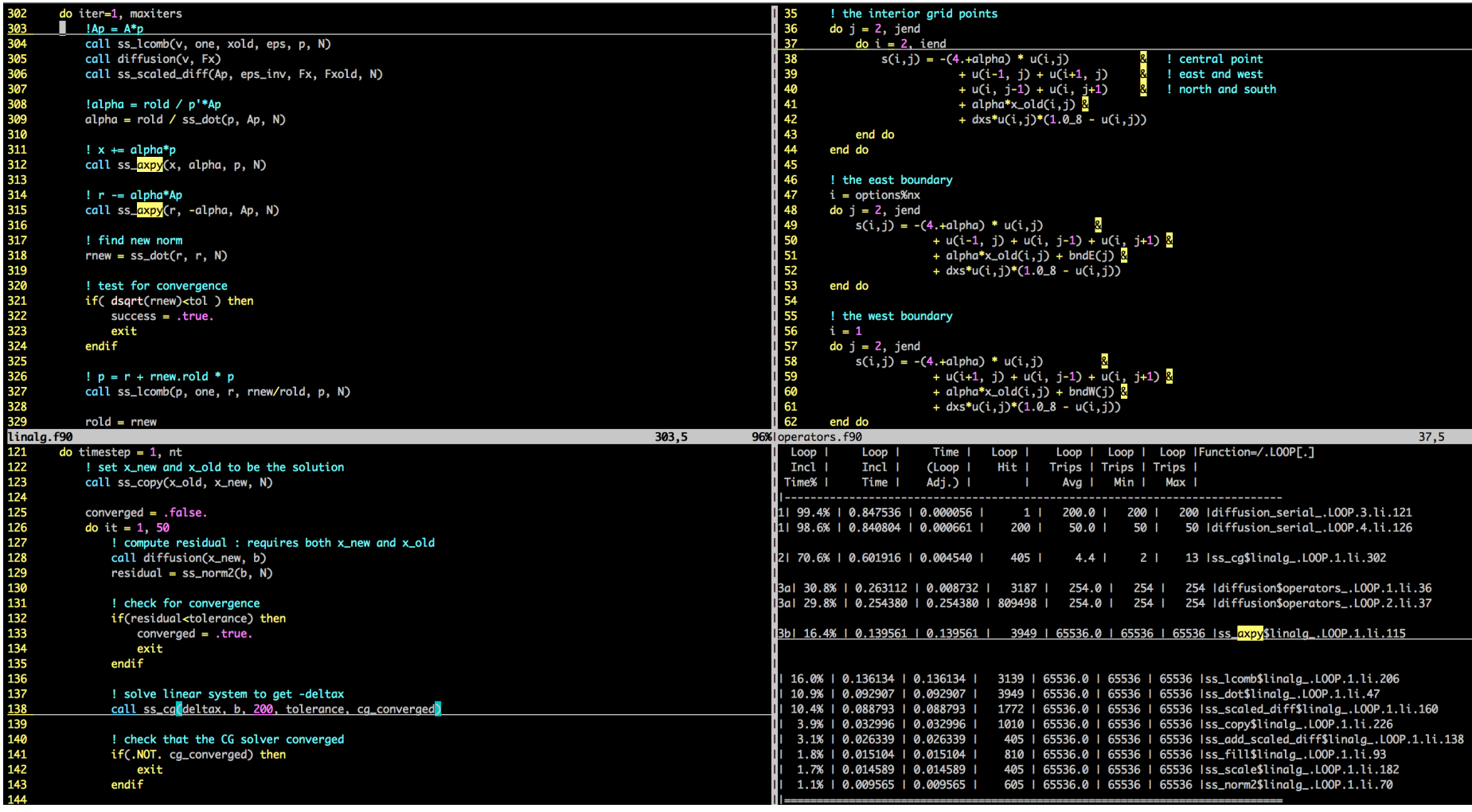
Reveal

serial miniapp + reveal

Compiling with the tool (CCE compiler only)

- Cray reveal can help you add OpenMP directives in the src:
- module load perftools
- make clean; make FFLAGS="-O3 -eZ `-hpl=reveal.pl`"
- no need to run
- reveal reveal.pl # requires X

serial miniapp + reveal: loops



```
302     do iter=1, maxiters
303        !Ap = A*p
304        call ss_lcomb(v, one, xold, eps, p, N)
305        call diffusion(v, Fx)
306        call ss_scaled_diff(Ap, eps_inv, Fx, Fxold, N)
307
308        !alpha = rold / p'*Ap
309        alpha = rold / ss_dot(p, Ap, N)
310
311        ! x += alpha*p
312        call ss_axpy(x, alpha, p, N)
313
314        ! r -= alpha*Ap
315        call ss_axpy(r, -alpha, Ap, N)
316
317        ! find new norm
318        rnew = ss_dot(r, r, N)
319
320        ! test for convergence
321        if( dsqrt(rnew)<tol ) then
322            success = .true.
323            exit
324        endif
325
326        ! p = r + rnew.rold * p
327        call ss_lcomb(p, one, r, rnew/rold, p, N)
328
329        rold = rnew
```

```
35      ! the interior grid points
36      do j = 2, jend
37         do i = 2, iend
38            s(i,j) = -(4.+alpha) * u(i,j)          &   ! central point
39                     + u(i-1, j) + u(i+1, j)       &   ! east and west
40                     + u(i, j-1) + u(i, j+1)       &   ! north and south
41                     + alpha*x_old(i,j) &
42                     + dxs*u(i,j)*(1.0_8 - u(i,j))
43         end do
44      end do
45
46      ! the east boundary
47      i = options%nx
48      do j = 2, jend
49          s(i,j) = -(4.+alpha) * u(i,j)        &
50                   + u(i-1, j) + u(i, j-1) + u(i, j+1) &
51                   + alpha*x_old(i,j) + bndE(j) &
52                   + dxs*u(i,j)*(1.0_8 - u(i,j))
53      end do
54
55      ! the west boundary
56      i = 1
57      do j = 2, jend
58          s(i,j) = -(4.+alpha) * u(i,j)        &
59                   + u(i+1, j) + u(i, j-1) + u(i, j+1) &
60                   + alpha*x_old(i,j) + bndW(j) &
61                   + dxs*u(i,j)*(1.0_8 - u(i,j))
62      end do
```

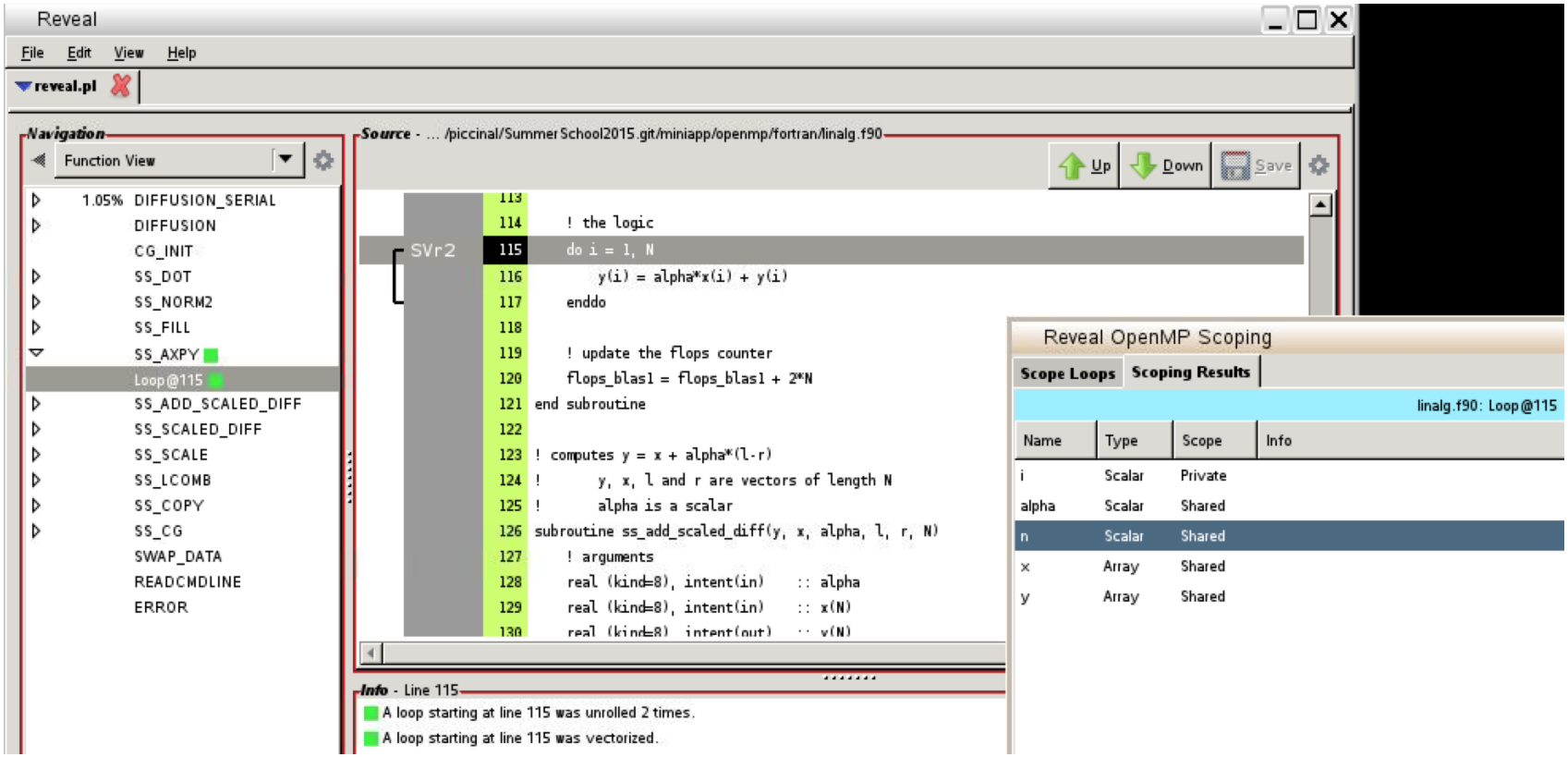
linalg.f90 303,5 96%

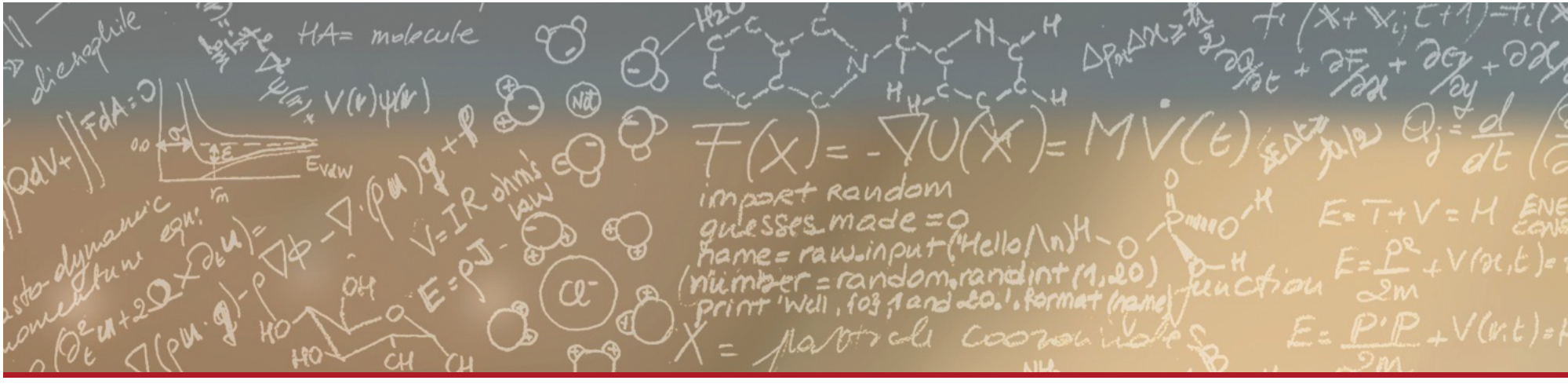
operators.f90 37,5

```
121     do timestep = 1, nt
122         ! set x_new and x_old to be the solution
123         call ss_copy(x_old, x_new, N)
124
125         converged = .false.
126         do it = 1, 50
127             ! compute residual : requires both x_new and x_old
128             call diffusion(x_new, b)
129             residual = ss_norm2(b, N)
130
131             ! check for convergence
132             if(residual<tolerance) then
133                 converged = .true.
134                 exit
135             endif
136
137             ! solve linear system to get -deltax
138             call ss_cg(deltax, b, 200, tolerance, cg_converged)
139
140             ! check that the CG solver converged
141             if(.NOT. cg_converged) then
142                 exit
143             endif
144
```

```
 Loop |    Loop |    Time |   Loop |    Loop |  Loop |  Loop |Function=/.LOOP[.]
 Incl |    Incl |   (Loop |    Hit |   Trips | Trips | Trips |
 Time% |    Time |    Adj.) |        |     Avg |   Min |   Max |
-------------------------------------------------------------------------------
1| 99.4% | 0.847536 | 0.000056 |      1 |   200.0 |   200 |   200 |diffusion_serial_.LOOP.3.li.121
1| 98.6% | 0.840804 | 0.000661 |    200 |    50.0 |    50 |    50 |diffusion_serial_.LOOP.4.li.126

2| 70.6% | 0.601916 | 0.004540 |    405 |     4.4 |     2 |    13 |ss_cg$linalg_.LOOP.1.li.302

3a| 30.8% | 0.263112 | 0.008732 |   3187 |   254.0 |   254 |   254 |diffusion$operators_.LOOP.1.li.36
3a| 29.8% | 0.254380 | 0.254380 | 809498 |   254.0 |   254 |   254 |diffusion$operators_.LOOP.2.li.37

3b| 16.4% | 0.139561 | 0.139561 |   3949 | 65536.0 | 65536 | 65536 |ss_axpy$linalg_.LOOP.1.li.115

| 16.0% | 0.136134 | 0.136134 |   3139 | 65536.0 | 65536 | 65536 |ss_lcomb$linalg_.LOOP.1.li.206
| 10.9% | 0.092907 | 0.092907 |   3949 | 65536.0 | 65536 | 65536 |ss_dot$linalg_.LOOP.1.li.47
| 10.4% | 0.088793 | 0.088793 |   1772 | 65536.0 | 65536 | 65536 |ss_scaled_diff$linalg_.LOOP.1.li.160
|  3.9% | 0.032996 | 0.032996 |   1010 | 65536.0 | 65536 | 65536 |ss_copy$linalg_.LOOP.1.li.226
|  3.1% | 0.026339 | 0.026339 |    405 | 65536.0 | 65536 | 65536 |ss_add_scaled_diff$linalg_.LOOP.1.li.138
|  1.8% | 0.015104 | 0.015104 |    810 | 65536.0 | 65536 | 65536 |ss_fill$linalg_.LOOP.1.li.93
|  1.7% | 0.014589 | 0.014589 |    405 | 65536.0 | 65536 | 65536 |ss_scale$linalg_.LOOP.1.li.182
|  1.1% | 0.009565 | 0.009565 |    605 | 65536.0 | 65536 | 65536 |ss_norm2$linalg_.LOOP.1.li.70
```

serial miniapp + reveal: scoping





Thank you for your attention.