Basic Debugging

October 10, 2012

NASA Advanced Supercomputing Division

Outline

- Code Porting vs. Debugging
- Debugging with Intel Compiler Flags
  - Case study #1: -g -traceback -check -fpe0
  - Case study #2: non-deterministic behavior
  - Case study #3: incorrect algorithm, sensitivity to round-off
- Debugging with TotalView
  - Getting started with TotalView on Pleiades
  - Simple navigation with the GUI
  - Case study #4: program hang
  - Where to get more information on using TotalView

Question? Use the Webex chat facility to ask the Host

Code Porting vs. Debugging

Porting -- Getting a code, generally from somewhere else, to compile/build and generate "expected" results on the target machine. If the results are unexpected, then:

1. Compile with -fp-model precise
2. Lower compiler optimization: try compiling with -O1 or -O2 (default for Intel)
3. Try a different combination of compilers and libraries

There are 30+ compilers, 20+ MPI libraries, 6 NetCDF and 15 HDF libraries on Pleiades!

- Debugging in this webinar refers to debugging codes written in Fortran, C or C++
- Debugging may be necessary as part of the porting process

  Why? Because you are getting <u>wrong</u> or <u>different</u> or no results

  - The first two of these could be a machine precision issue, or a compiler optimization issue (See "Porting" above)
  - The 3rd could be due to a program hang, segfault, abort, etc.
- Debugging may be necessary if you are writing new code or modified an existing one

Question? Use the Webex chat facility to ask the Host

Debugging with Intel Compiler Flags

For Fortran use: -g -traceback -check -fpe0

For C/C++ use: -g -traceback

-g

- Tells compiler to generate full debugging information in the object file (.o file)
- Changes the default optimization to -O0, so need to explicitly add -O2 if no optimization level was previously specified
- Always compile with -g when using debuggers (i.e., TotalView)

-traceback

- Provides traceback information with source file, routine name, and line number when a severe error occurs at run time

```
forrtl: error (73): floating divide by zero
Image              PC                 Routine            Line        Source
buggy              0000000000403144   sub1_                    24    buggy.f
buggy              0000000000402ECF   MAIN__                   13    buggy.f
buggy              0000000000402C0C   Unknown             Unknown    Unknown
libc.so.6          00007FFFECF2FBC6   Unknown             Unknown    Unknown
buggy              0000000000402B09   Unknown             Unknown    Unknown
```


Question? Use the Webex chat facility to ask the Host

Debugging with Intel Compiler Flags (2 of 3)

-check (same as -check all)

- checks for array bounds violation (same as -check bounds)
  - Example: dimension a(100) and the code uses a(101) = …
- checks for use of uninitialized variables (same as -check uninit)
  - Caution: the checking is very limited in scope
- checks for format, output_conversion, pointers, etc. … generally, unlikely to be the culprit
- Important: -check causes the program to run slow! Leave off after debugging

-fpe0

- traps floating-point exceptions, i.e., divide-by-zero, sqrt of negative, etc.
- when compiling with -fpe0, **all** source files need to be compiled with this flag (or with -fp-speculation=off) to avoid false-positives
  - Example:

```
if (z .ne. 0.0) then
    y = 1/z
else
    y = 1 + z
endif
```

both branches are executed simultaneously and one branch is discarded after evaluating the if conditional

Question? Use the Webex chat facility to ask the Host

Debugging with Intel Compiler Flags (3 of 3)

Example of array bounds violation message

```
forrtl: severe (408): fort: (2): Subscript #1 of the array Y has value 101 which
is greater than the upper bound of 100
```

Image	PC	Routine	Line	Source
buggy	0000000000046AFCA	Unknown	Unknown	Unknown
buggy	0000000000469AC6	Unknown	Unknown	Unknown
buggy	0000000000421DE0	Unknown	Unknown	Unknown
buggy	0000000000404B6E	Unknown	Unknown	Unknown
buggy	0000000000405091	Unknown	Unknown	Unknown
buggy	00000000004033D9	sub1_	27	buggy.f
buggy	0000000000402F82	MAIN__	13	buggy.f
buggy	0000000000402C0C	Unknown	Unknown	Unknown
libc.so.6	00007FFFECF2FBC6	Unknown	Unknown	Unknown
buggy	0000000000402B09	Unknown	Unknown	Unknown

Question? Use the Webex chat facility to ask the Host

Case Study #1: Program buggy

```
program buggy
parameter (nmax = 100)
real (kind=8) :: a(nmax), b(nmax), c

a = 2.0   ! sets all elements of a() to 2.0
b = 3.0   ! sets all elements of b() to 3.0

print *, 'a(1) + a(2) = ', a(1) + a(2)
print *, 'b(1) + b(2) = ', b(1) + b(2)

b(3) = a(3) + c

call sub1(nmax,a,b,c)

print *, 'after calling sub1'
print *, 'a(1) + a(2) = ', a(1) + a(2)
print *, 'b(1) + b(2) = ', b(1) + b(2)
end

subroutine sub1(n,x,y,z)
real (kind = 8) :: x(n), y(n), z

y(3) = x(3) + z  ! note uninit doesn't trap here
y(4) = 1.0/z

do i = 10,200
  y(i) = dfloat(i)
  x(i) = 1.0/dfloat(i)
enddo

end
```


Question? Use the Webex chat facility to ask the Host

Case Study #1: Initial run and debug

```
pfe8> ifort -o buggy buggy.f
pfe8> buggy
 a(1) + a(2) =    4.00000000000000
 b(1) + b(2) =    6.00000000000000
 after calling sub1
 a(1) + a(2) =    4.00000000000000
 b(1) + b(2) =   1.970491166763735E-002
< ...program hangs >
```

Note wrong answer after calling sub1 and program hang.

```
pfe8> ifort -o buggy -g -traceback -check -fpe0 buggy.f
pfe8> buggy
 a(1) + a(2) =    4.00000000000000
 b(1) + b(2) =    6.00000000000000
forrtl: severe (193): Run-Time Check Failure. The variable 'buggy_$C' is being u
sed without being defined
Image              PC                Routine            Line        Source

buggy              000000000046B20A  Unknown               Unknown  Unknown
buggy              0000000000469D06  Unknown               Unknown  Unknown
buggy              0000000000422020  Unknown               Unknown  Unknown
buggy              0000000000404DAE  Unknown               Unknown  Unknown
buggy              0000000000405C88  Unknown               Unknown  Unknown
buggy              0000000000402F6F  MAIN__                     11  buggy.f
buggy              0000000000402C0C  Unknown               Unknown  Unknown
libc.so.6          00007FFFECF2FBC6  Unknown               Unknown  Unknown
buggy              0000000000402B09  Unknown               Unknown  Unknown
pfe8>
```

forrtl is the Fortran Runtime Library (RTL)

Note variable name and offending line number in the traceback.

Question? Use the Webex chat facility to ask the Host

Case Study #1: Modify code and re-run

```
 1        program buggy
 2        parameter (nmax = 100)
 3        real (kind=8) :: a(nmax), b(nmax), c
 4
 5        a = 2.0   ! sets all elements of a() to 2.0
 6        b = 3.0   ! sets all elements of b() to 3.0
 7
 8        print *, 'a(1) + a(2) = ', a(1) + a(2)
 9        print *, 'b(1) + b(2) = ', b(1) + b(2)
10
11 !      b(3) = a(3) + c
12
13        call sub1(nmax,a,b,c)
14
```

Need to assign a value to “c” before use, but for now just comment line 11 out and continue.

```
pfe8> vi buggy.f
pfe8> !ifort
ifort -o buggy -g -traceback -check -fpe0 buggy.f
pfe8> buggy
 a(1) + a(2) =    4.00000000000000
 b(1) + b(2) =    6.00000000000000
forrtl: error (73): floating divide by zero
Image              PC                Routine            Line        Source

buggy              0000000000403423  sub1_                    24  buggy.f
buggy              0000000000402F82  MAIN__                   13  buggy.f
buggy              0000000000402C0C  Unknown            Unknown  Unknown
libc.so.6          00007FFFECF2FBC6  Unknown            Unknown  Unknown
buggy              0000000000402B09  Unknown            Unknown  Unknown
Abort (core dumped)
pfe8>
```

Recompile and re-run. The code now aborts with a divide-by-zero (caught by -fpe0) at line 24.

Question? Use the Webex chat facility to ask the Host

Case Study #1: Modify code and re-run

```
1        program buggy
2        parameter (nmax = 100)
3        real (kind=8) :: a(nmax), b(nmax), c
4
5        a = 2.0   ! sets all elements of a() to 2.0
6        b = 3.0   ! sets all elements of b() to 3.0
7
8        print *, 'a(1) + a(2) = ', a(1) + a(2)
9        print *, 'b(1) + b(2) = ', b(1) + b(2)
10
11 !     b(3) = a(3) + c
12
13       call sub1(nmax,a,b,c)
14
15       print *, 'after calling sub1'
16       print *, 'a(1) + a(2) = ', a(1) + a(2)
17       print *, 'b(1) + b(2) = ', b(1) + b(2)
18       end
19
20       subroutine sub1(n,x,y,z)
21       real (kind = 8) :: x(n), y(n), z
22
23       y(3) = x(3) + z  ! note uninit doesn't trap here
24       y(4) = 1.0/z
25
26       do i = 10,200
27         y(i) = dfloat(i)
28         x(i) = 1.0/dfloat(i)
29       enddo
30
31       end
```

"c" wasn't assigned a value, so it turned out to be 0.0 (but you can't always depend on uninitialized variables having a value of zero!).

"c" was passed into sub1, so the dummy argument "z" has a value of 0 -- triggering the divide-by-zero at line 24. But WAIT ... note that the RTL didn't catch that "z" was being used at line 23 even though it hasn't been initialized!

Question? Use the Webex chat facility to ask the Host

Case Study #1: Modify code and re-run

```
20        subroutine sub1(n,x,y,z)
21        real (kind = 8) :: x(n), y(n), z
22
23        y(3) = x(3) + z  ! note uninit doesn't trap here
24 !      y(4) = 1.0/z
25
26        do i = 10,200
27          y(i) = dfloat(i)
28          x(i) = 1.0/dfloat(i)
29        enddo
30
31        end
```

Need to fix the divide-by-zero, but for now just comment it out and continue.

```
pfe8> !ifort
ifort -o buggy -g -traceback -check -fpe0 buggy.f
pfe8> buggy
 a(1) + a(2) =    4.00000000000000
 b(1) + b(2) =    6.00000000000000
forrtl: severe (408): fort: (2): Subscript #1 of the array Y has value 101 which
 is greater than the upper bound of 100

Image              PC                Routine            Line        Source

buggy              000000000046B10A  Unknown               Unknown  Unknown
buggy              0000000000469C06  Unknown               Unknown  Unknown
buggy              0000000000421F20  Unknown               Unknown  Unknown
buggy              0000000000404CAE  Unknown               Unknown  Unknown
buggy              00000000004051D1  Unknown               Unknown  Unknown
buggy              0000000000403515  sub1_                     27   buggy.f
buggy              0000000000402F82  MAIN__                    13   buggy.f
buggy              0000000000402C0C  Unknown               Unknown  Unknown
libc.so.6          00007FFFECF2FBC6  Unknown               Unknown  Unknown
buggy              0000000000402B09  Unknown               Unknown  Unknown
pfe8>
```

Arrays a and b, which are passed into sub1 as x and y, were declared with dimension NMAX=100.

If y were a multi-dimensional array, the forrtl complaint could point to subscript #2 or #3, for example.

Question? Use the Webex chat facility to ask the Host

Case Study #1: Modify code and re-run

```
20        subroutine sub1(n,x,y,z)
21        real (kind = 8) :: x(1), y(1), z
22
23        y(3) = x(3) + z  ! note uninit doesn't trap here
24 !      y(4) = 1.0/z
25
26        do i = 10,200
27          y(i) = dfloat(i)
28          x(i) = 1.0/dfloat(i)
29        enddo
30
31        end
```

Need to change the limits of the DO loop at line 26 or increase NMAX, but let's see what happens when the arrays x and y are dimensioned 1 (or *) as in many legacy codes. This is perfectly legal Fortran, btw.

```
pfe8> !ifort
ifort -o buggy -g -traceback -check -fpe0 buggy.f
pfe8> buggy
 a(1) + a(2) =    4.00000000000000
 b(1) + b(2) =    6.00000000000000
pfe8>
```

The -check flag no longer catches array bounds violation on arrays x and y and the program ends without errors, but the result is unexpected.

```
pfe8> !ifort
ifort -o buggy -g -traceback -check -fpe0 buggy.f
pfe8> buggy
 a(1) + a(2) =    4.00000000000000
 b(1) + b(2) =    6.00000000000000
 after calling sub1
 a(1) + a(2) =    4.00000000000000
 b(1) + b(2) =    6.00000000000000
pfe8>
```

This is the correct behavior after the remaining array bounds violation bugs have been removed.

Question? Use the Webex chat facility to ask the Host

Case Study #1: Take home

- If arrays are dimension 1 or *, for example, a(1), x(1,nj), then array bounds checking on those arrays is ineffective.
- Checking of uninitialized variables is limited in scope.
  (However, check back for the next webinar on "Uninit," which provides a technique for trapping all uninitialized variables)
- Debugging with compiler debug options is useful for catching most common bugs, but **not** all bugs. So, just because a code has passed a "health check" using the compiler debug options does not mean it's bug-free.
- Debugging with compiler flags is an iterative process:
  compile → run → modify → compile → run → ...
- To streamline the iterative process, get an interactive PBS session:
  qsub -I -q devel -lselect=NN:ncpus=YY,walltime=2:00:00
  and make successive runs in the same PBS session.
  To avoid typing all the commands in runscript, just make it executable (chmod u+x runscript) and run interactively:
  ./runscript

Question? Use the Webex chat facility to ask the Host

Case Study #2: “Magic Trick”

```
      program main
! compile with pgf90
      integer m,n
      m = 5
      n = 7

      call switcha(m, n, .True.)
      call printa(m, n, .True.)

      end

      subroutine switcha(m, n, iflag)
      logical iflag
      integer m,n,k

      if (iflag) then
        k=m
        m=n
        n=k
        iflag = .False.
      endif

      end

      subroutine printa(m, n, iflag)
      logical iflag
      if (iflag) then
        print *, 'm = ',m
      else
        print *, 'n = ',n
      endif
      end
```


Question? Use the Webex chat facility to ask the Host

Case Study #2: Non-deterministic behavior

```
cfe2> ifort -o ex1 tv_ex1.f
cfe2> ex1
 n =            5
cfe2> module list
Currently Loaded Modulefiles:
  1) intel-comp.11.1.072
cfe2> module switch intel-comp.11.1.072 intel-comp.10.1.013
cfe2> !ifort
ifort -o ex1 tv_ex1.f
cfe2> ex1
 m =            7
```

Different versions of ifort give different answers on Columbia.

Code aborts with a segfault with Intel compiler and gives one of the two Columbia results with PGI compiler.

```
pfe8> ifort -o ex1 tv_ex1.f
pfe8> ex1
forrtl: severe (174): SIGSEGV, segmentation fault occurred
Image              PC                Routine            Line        Source

ex1                0000000000402C8E  Unknown               Unknown  Unknown
ex1                0000000000402C0C  Unknown               Unknown  Unknown
libc.so.6          00007FFFED133BC6  Unknown               Unknown  Unknown
ex1                0000000000402B09  Unknown               Unknown  Unknown
pfe8> module purge
pfe8> module load comp-pgi/12.5
pfe8> pgf90 -o ex1_pg tv_ex1.f
pfe8> ex1_pg
 n =            5
```

Take home:

- Bugs can cause different results with different compilers, different machines, different compiler options, etc.
- Just because you get the same results doesn't mean the code is bug-free!

Question? Use the Webex chat facility to ask the Host

Case Study #2: Bug Revealed

```
pfe8> ifort -o ex1_dbg -g -traceback -check -fpe0 tv_ex1.f
pfe8> ex1_dbg
forrtl: severe (174): SIGSEGV, segmentation fault occurred
Image              PC                Routine            Line        Source

ex1_dbg            0000000000402D24  switcha_                 20  tv_ex1.f
ex1_dbg            0000000000402C92  MAIN__                    6  tv_ex1.f
ex1_dbg            0000000000402C0C  Unknown               Unknown  Unknown
libc.so.6          00007FFFECF2FBC6  Unknown               Unknown  Unknown
ex1_dbg            0000000000402B09  Unknown               Unknown  Unknown
```

The segfault with ifort on Pleiades is a clue that there is a bug in the code, so compile with debug flags.

```
 1        program main
 2        integer m,n
 3        m = 5
 4        n = 7
 5
 6        call switcha(m, n, .True.)
 7
 8        call printa(m, n, .True.)
 9
10        end
11
12        subroutine switcha(m, n, iflag)
13        logical iflag
14        integer m,n,k
15
16        if (iflag) then
17          k=m
18          m=n
19          n=k
20          iflag = .False.
21        endif
```

Fortran passes by address into routines, so iflag in switcha contains the memory address of the literal constant .True., and, thus, it cannot be overwritten at line 20.

Question? Use the Webex chat facility to ask the Host

Case Study #2: Simpler "Magic Trick"

```
pfe8> cat tv_ex2.f
      program main
      integer i,j

      call first(1,j)
      call second(1)
      end

      subroutine first(i,j)
      integer i,j
      i = 5
      j = 7
      end

      subroutine second(k)
      integer k
      print *, 'one = ',k
      end
pfe8> pgf90 -o ex2_pg tv_ex2.f
pfe8> ex2_pg
 one =              5
```


Question? Use the Webex chat facility to ask the Host

Case Study #3: Improper algorithm or sensitivity to round-off error

Write a program to evaluate:

$$\lim_{x \to \infty} \left\{ \sqrt{x^2 + 3x} - \sqrt{x^2 + 2x} \right\}$$

```
      program main
! estimate limit as x approaches infinity
      real (kind=8) :: x, y

      x = 1.0d0
      do i = 1,30
         y = sqrt(x*x + 3.248d0*x) - sqrt(x*x + 2.d0*x)
         print *, 'x,y = ',x,y
         x = x*10.0d0
      enddo

      end
```

Note: 3 has been changed to 3.248 to show loss of significant figures in the results.

Question? Use the Webex chat facility to ask the Host

Case Study #3: Results

```
pfe8> ifort -o xinf xinf.f
pfe8> xinf
 x,y =    1.00000000000000        0.329016876909243
 x,y =    10.0000000000000        0.555544505847205
 x,y =    100.000000000000        0.615973640458648
 x,y =    1000.00000000000        0.623182949835950
 x,y =    10000.0000000000        0.623918147612130
 x,y =    100000.000000000        0.623991813277826
 x,y =    1000000.00000000        0.623999181319959
 x,y =    10000000.0000000        0.623999917879701
 x,y =    100000000.000000        0.623999983072281
 x,y =    1000000000.00000        0.623999953269958
 x,y =    10000000000.0000        0.623998641967773
 x,y =    100000000000.000        0.624008178710938
 x,y =    1000000000000.00        0.624023437500000
 x,y =    10000000000000.0        0.623046875000000
 x,y =    100000000000000.        0.640625000000000
 x,y =   1.000000000000000E+015   0.625000000000000
 x,y =   1.000000000000000E+016   0.000000000000000E+000
 x,y =   1.000000000000000E+017   0.000000000000000E+000
 x,y =   1.000000000000000E+018   0.000000000000000E+000
 x,y =   1.000000000000000E+019   0.000000000000000E+000
 x,y =   1.000000000000000E+020   0.000000000000000E+000
 x,y =   1.000000000000000E+021   0.000000000000000E+000
 x,y =   1.000000000000000E+022   0.000000000000000E+000
 x,y =   9.999999999999999E+022   0.000000000000000E+000
 x,y =   1.000000000000000E+024   0.000000000000000E+000
 x,y =   9.999999999999999E+024   0.000000000000000E+000
 x,y =   9.999999999999999E+025   0.000000000000000E+000
 x,y =   9.999999999999999E+026   0.000000000000000E+000
 x,y =   1.000000000000000E+028   0.000000000000000E+000
 x,y =   9.999999999999999E+028   0.000000000000000E+000
```


Question? Use the Webex chat facility to ask the Host

Case Study #3: Take home

Analytical answer: $$\left\{\sqrt{x^2+3x}-\sqrt{x^2+2x}\right\}\frac{\left(\sqrt{x^2+3x}+\sqrt{x^2+2x}\right)}{\left(\sqrt{x^2+3x}+\sqrt{x^2+2x}\right)}=\frac{x}{\left(\sqrt{x^2+3x}+\sqrt{x^2+2x}\right)}$$

$$\lim_{x\to\infty}\left\{\sqrt{x^2+3x}-\sqrt{x^2+2x}\right\}=1/2 \qquad \lim_{x\to\infty}\left\{\sqrt{x^2+3.248x}-\sqrt{x^2+2x}\right\}=0.624$$

- Note that for $x > 10^{16}$ the 3.248x and 2x terms under the square root get dropped off compared to x^2.
- The correct answer, 0.624 to 15 significant figures, is only obtained for a very narrow range in the value of x (if at all) for this program.
- If the formula used in the program is changed to the one above on the right hand side, then the correct answer can be obtained for any $x > 10^{16}$.
- Is the issue here one of round-off errors or incorrect formula?
- Part of debugging is figuring out when the answers are as good as they are going to get due to sensitivities to round-off error, unstable algorithm, etc.

Question? Use the Webex chat facility to ask the Host

Debugging with TotalView

Question? Use the Webex chat facility to ask the Host

Getting started with TotalView on Pleiades

1. Make sure that you can display X11 graphics from pfe
   - log in with ssh -X or add “ForwardX11 yes” to your .ssh/config file on your workstation
   - echo $DISPLAY should show some setting for the DISPLAY environment set by ssh
   - test by running xclock
   - may need to use VNC if interactive response is very slow
2. Compile your code with the -g compiler flag
3. Submit an interactive PBS job and forward your DISPLAY environment
   - qsub -I -v DISPLAY -q devel -lselect=N:ncpus=XX,walltime=HH:MM:SS
4. Once the PBS job has started, load the totalview module if it is not automatically loaded from your .login or .cshrc file
   - module load totalview/8.9.2-1
5. Start totalview with either:

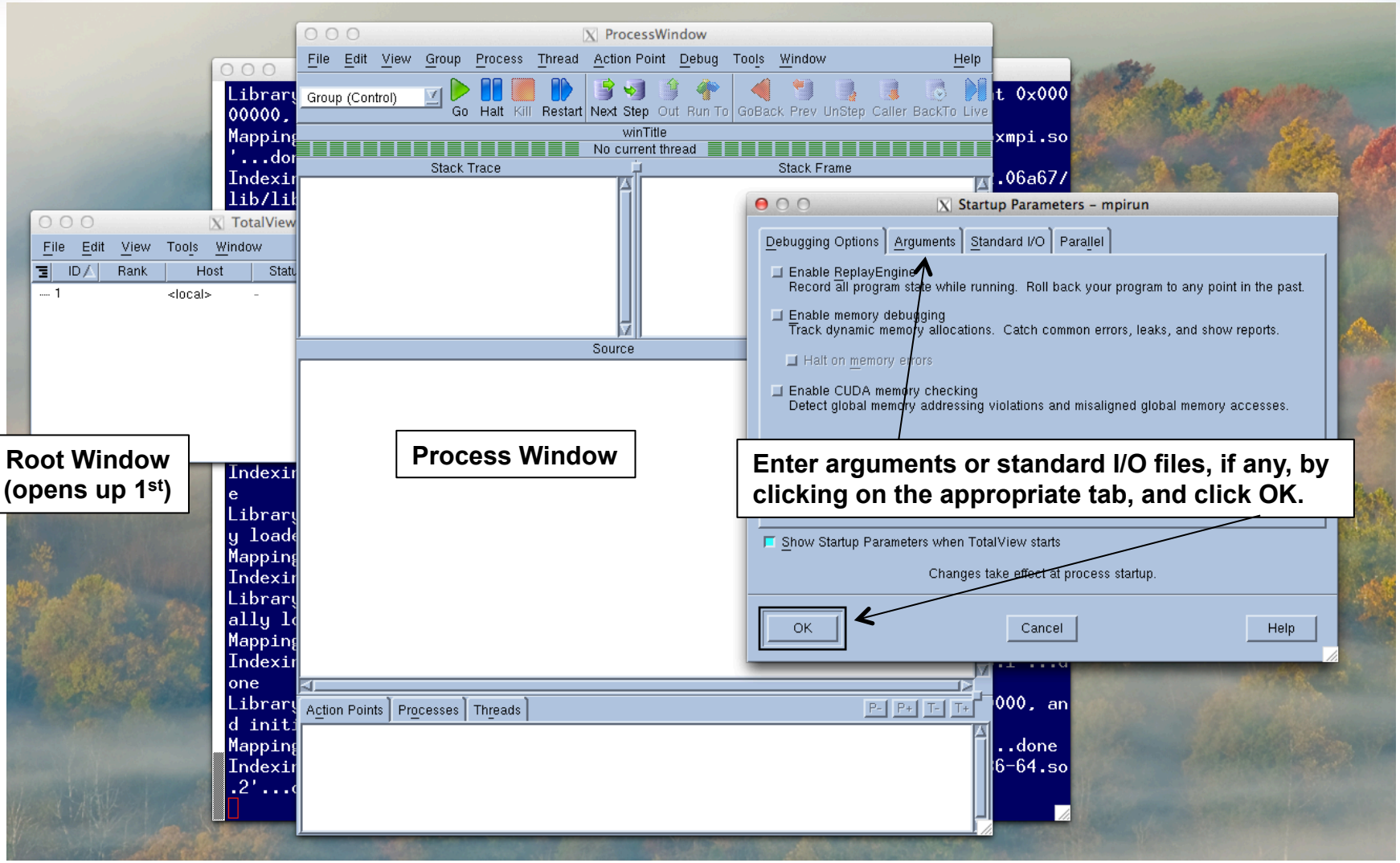
   totalview ./exe [-a args] for serial or OpenMP programs

   or

   mpiexec_mpt -tv -np <nprocs> ./exe for MPI or hybrid programs (assumes MPT)

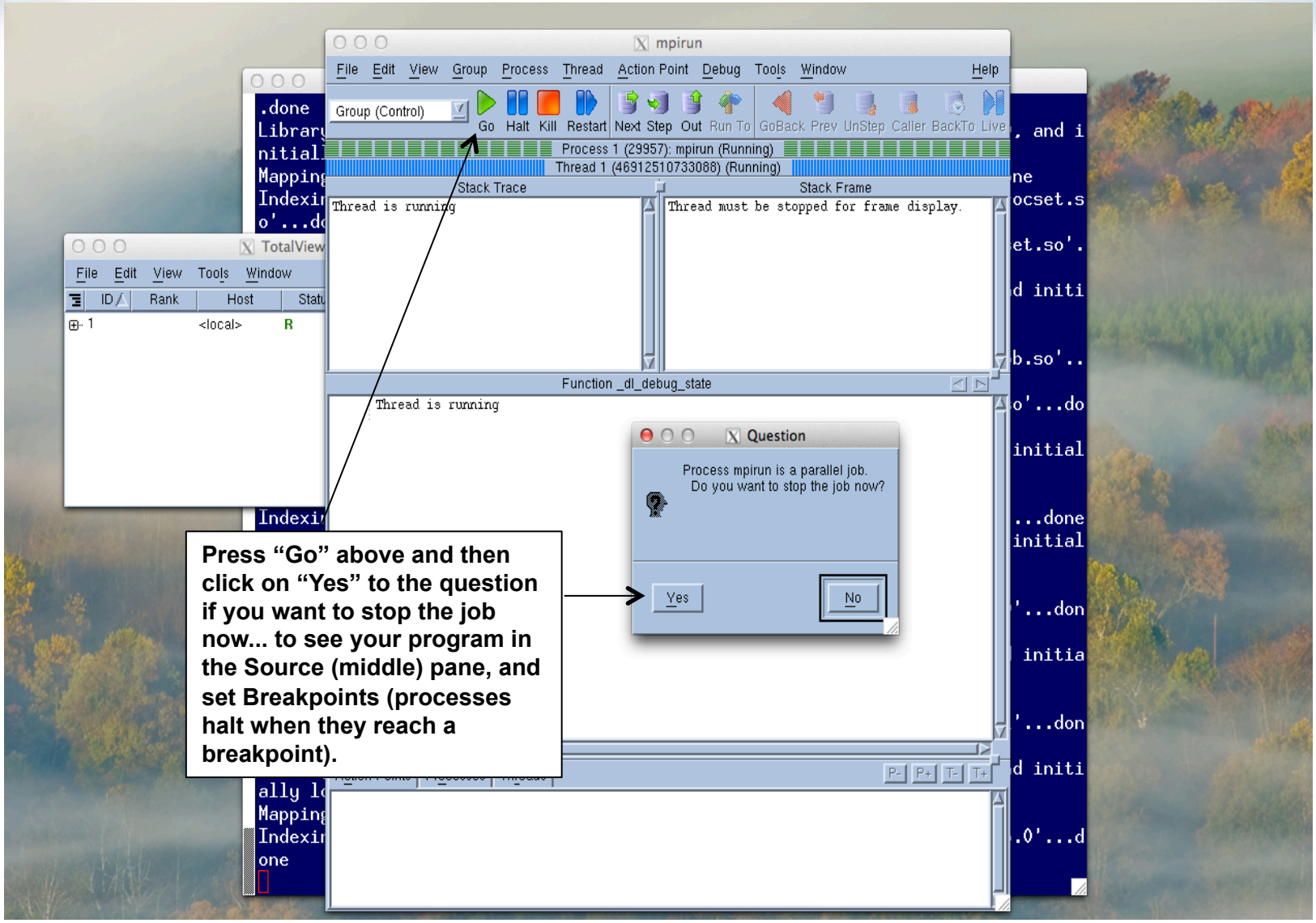
Question? Use the Webex chat facility to ask the Host

TotalView Start-up View



Question? Use the Webex chat facility to ask the Host

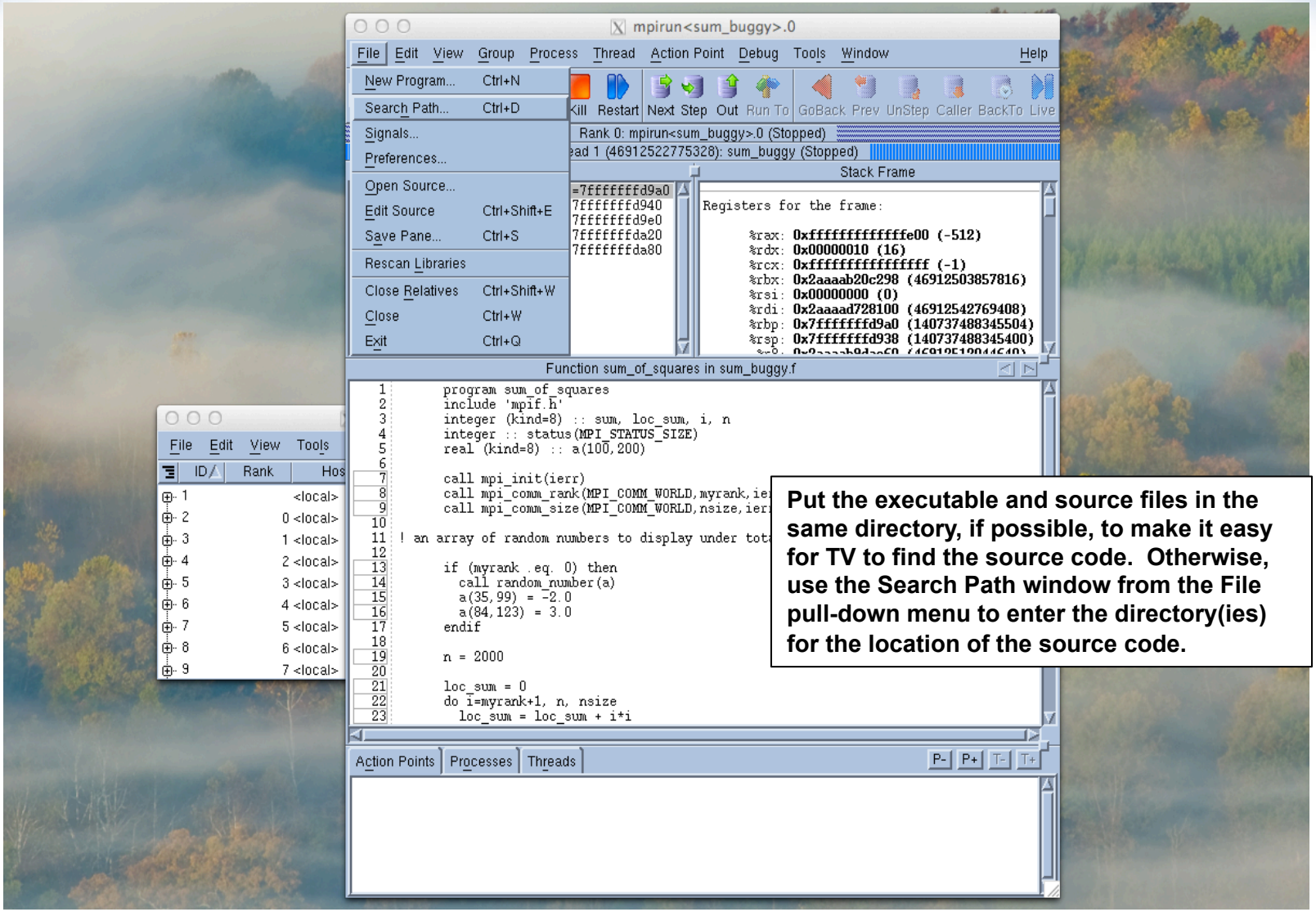
Simple Navigation with the GUI



Press "Go" above and then click on "Yes" to the question if you want to stop the job now... to see your program in the Source (middle) pane, and set Breakpoints (processes halt when they reach a breakpoint).

Question? Use the Webex chat facility to ask the Host

Search Path for Source Code



Put the executable and source files in the same directory, if possible, to make it easy for TV to find the source code. Otherwise, use the Search Path window from the File pull-down menu to enter the directory(ies) for the location of the source code.

Question? Use the Webex chat facility to ask the Host

Navigate in Process Window

Go -- starts the program running
Halt -- stops it
Kill -- kills all the processes
Restart -- restarts from the beginning

Next -- goes to the next line in the program even if it is a function or routine Step -- same as Next except that it will step into the function or routine
Out -- runs to the completion of the function or routine and steps out to the caller.

mpirun<sum_buggy>.0

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) | Go | Halt | Kill | Restart | Next | Step | Out | Run To | GoBack | Prev | UnStep | Caller | BackTo | Live

Rank 0: mpirun<sum_buggy>.0 (Stopped)

Thread 1 (46912522775328): sum_buggy (Stopped)

Stack Trace

```
    pthread_barrier_wait,  FP=7fffffffd9a0
C   MPI_SGI_barsync,       FP=7fffffffd940
C   MPI_SGI_init,          FP=7fffffffd9e0
    call_init,             FP=7fffffffda20
    _dl_init,              FP=7fffffffda80
```

Stack Frame

```
Registers for the frame:

      %rax: 0xffffffffffffffe00 (-512)
      %rdx: 0x00000010 (16)
      %rcx: 0xffffffffffffffff (-1)
      %rbx: 0x2aaaab20c298 (46912503857816)
      %rsi: 0x00000000 (0)
      %rdi: 0x2aaaad728100 (46912542769408)
      %rbp: 0x7fffffffd9a0 (140737488345504)
      %rsp: 0x7fffffffd938 (140737488345400)
```

Function sum_of_squares in sum_buggy.f

```
 1        program sum_of_squares
 2        include 'mpif.h'
 3        integer (kind=8) :: sum, loc_sum, i, n
 4        integer :: status(MPI_STATUS_SIZE)
 5        real (kind=8) :: a(100,200)
 6
 7        call mpi_init(ierr)
 8        call mpi_comm_rank(MPI_COMM_WORLD,myrank,ierr)
 9        call mpi_comm_size(MPI_COMM_WORLD,nsize,ierr)
10
11  ! an array of random numbers to display under totalview
12
13        if (myrank .eq. 0) then
14          call random_number(a)
15          a(35,99) = -2.0
16          a(84,123) = 3.0
17        endif
18
STOP      n = 2000
20
21        loc_sum = 0
22        do i=myrank+1, n, nsize
23          loc_sum = loc_sum + i*i
```

Action Points | Processes | Threads | P- | P+ | T- | T+

```
STOP  1  sum_buggy.f#19  sum_of_squares+0x191
```

Arrows navigate the view of the source code up or down the call graph.

Breakpoints can be set at any location containing a rectangular box by clicking on the box, or via the Action Point pull-down menu above.

Program is still halted at the beginning, click Go to run to the first breakpoint at line 19.

Question? Use the Webex chat facility to ask the Host

At first breakpoint

Process window for Rank 0

All processes at first breakpoint

Location of program counter shown by arrow and highlighted in yellow

Press P+ or P- to see other processes up or down in rank

mpirun<sum_buggy>.0

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) | Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live

Rank 0: mpirun<sum_buggy>.0 (At Breakpoint 1)

Thread 1 (46912522775328): sum_buggy (At Breakpoint 1)

Stack Trace

```
f90  sum_of_squares,     FP=7fffffffd8f0
     main,               FP=7fffffffd9c0
     __libc_start_main,  FP=7fffffffda70
     _start,             FP=7fffffffda80
```

Stack Frame

```
Function "sum_of_squares":
  No arguments.
Local variables:
  nsize:      24 (0x00000018)
  myrank:     0 (0x00000000)
  ierr:       0 (0x00000000)
  a:          (REAL*8(100,200))
  status:     (INTEGER*4(6))
  n:          0 (0x0000000000000000)
  i:          0 (0x0000000000000000)
  loc_sum:    0 (0x0000000000000000)
```

Function sum_of_squares in sum_buggy.f

```
 7        call mpi_init(ierr)
 8        call mpi_comm_rank(MPI_COMM_WORLD,myrank,ierr)
 9        call mpi_comm_size(MPI_COMM_WORLD,nsize,ierr)
10
11  ! an array of random numbers to display under totalview
12
13        if (myrank .eq. 0) then
14          call random_number(a)
15          a(35,99) = -2.0
16          a(84,123) = 3.0
17        endif
18
19        n = 2000
20
21        loc_sum = 0
22        do i=myrank+1, n, nsize
23          loc_sum = loc_sum + i*i
24        enddo
25
26        print *, "Rank ",myrank, "; partial sum = ", loc_sum
27
28        if (myrank .eq. 0) then
29          sum = loc_sum
```

Action Points | Processes | Threads P- P+ T- T+

```
1  sum_buggy.f#19  sum_of_squares+0x191
```

TotalView 8.9.2-1

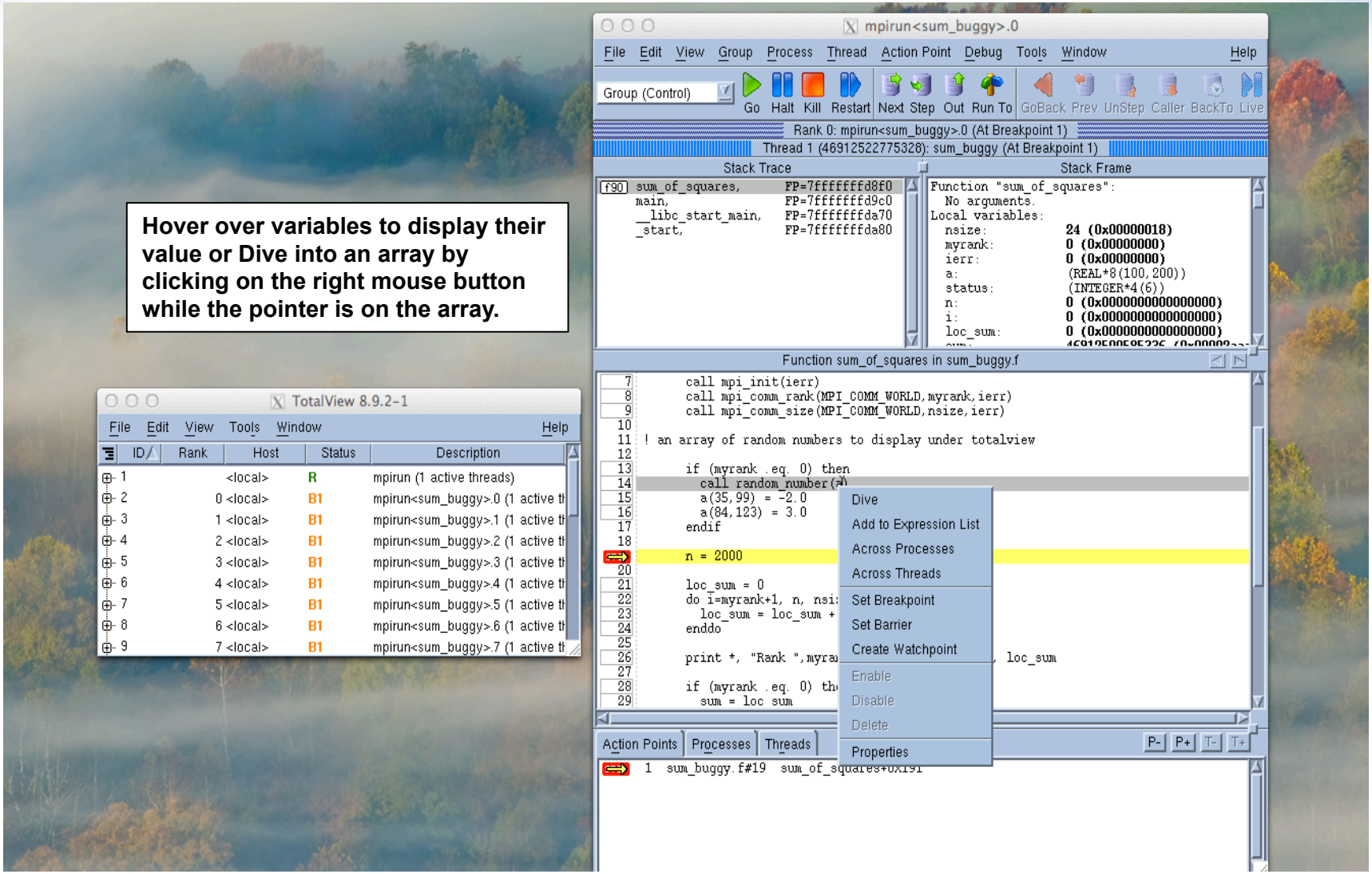
File Edit View Tools Window Help

ID	Rank	Host	Status	Description
1		<local>	R	mpirun (1 active threads)
2	0	<local>	B1	mpirun<sum_buggy>.0 (1 active th
3	1	<local>	B1	mpirun<sum_buggy>.1 (1 active th
4	2	<local>	B1	mpirun<sum_buggy>.2 (1 active th
5	3	<local>	B1	mpirun<sum_buggy>.3 (1 active th
6	4	<local>	B1	mpirun<sum_buggy>.4 (1 active th
7	5	<local>	B1	mpirun<sum_buggy>.5 (1 active th
8	6	<local>	B1	mpirun<sum_buggy>.6 (1 active th
9	7	<local>	B1	mpirun<sum_buggy>.7 (1 active th

Question? Use the Webex chat facility to ask the Host

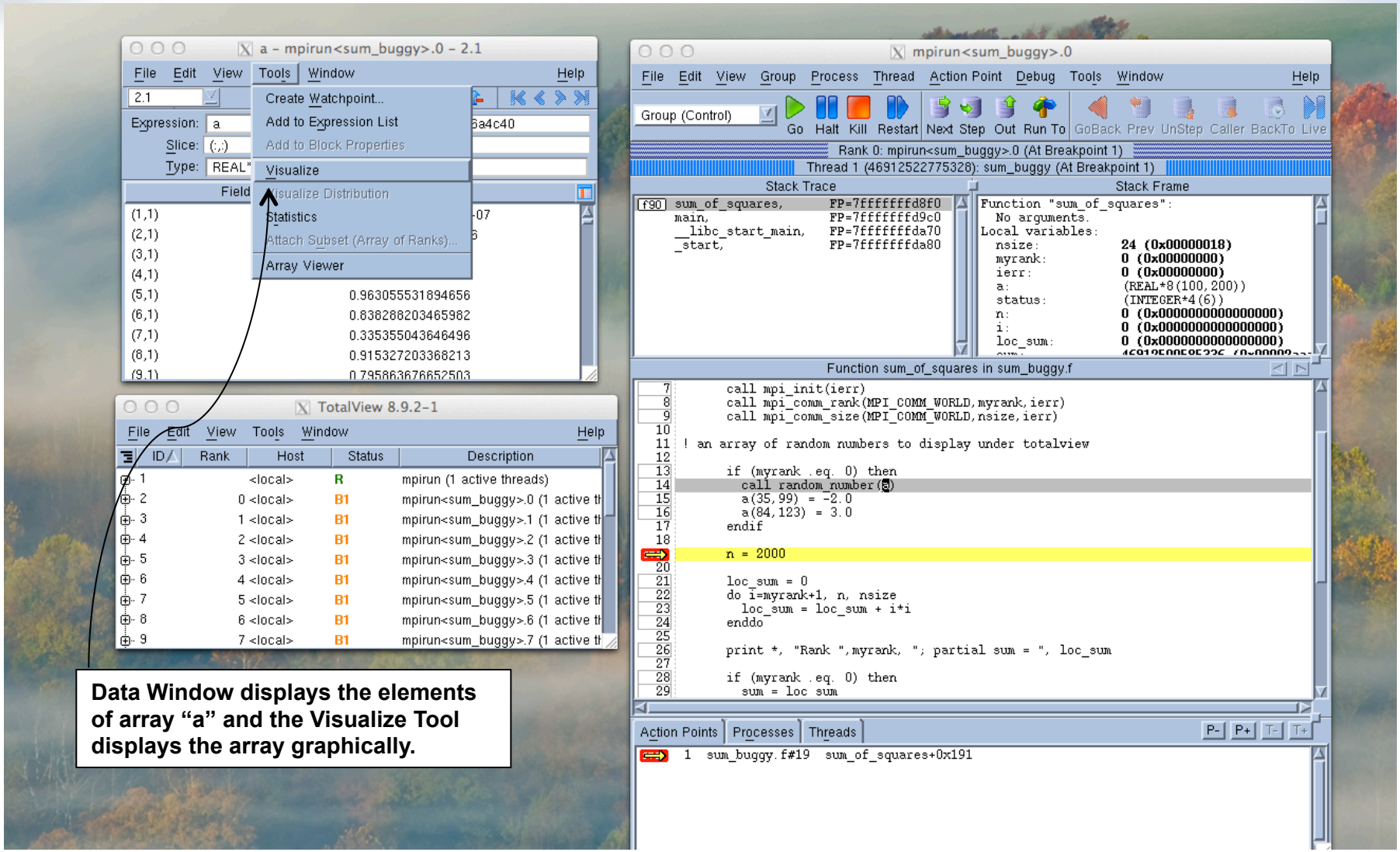
Diving

Hover over variables to display their value or Dive into an array by clicking on the right mouse button while the pointer is on the array.



Question? Use the Webex chat facility to ask the Host

Data Display



Data Window displays the elements of array "a" and the Visualize Tool displays the array graphically.

Question? Use the Webex chat facility to ask the Host

Visualizer, Data Slicer, and Filter

Filter which values to display

a - mpirun<sum_buggy>.0 - 2.1

File Edit View Tools Window Help

2.1

Expression: a Address: 0x006a4c40

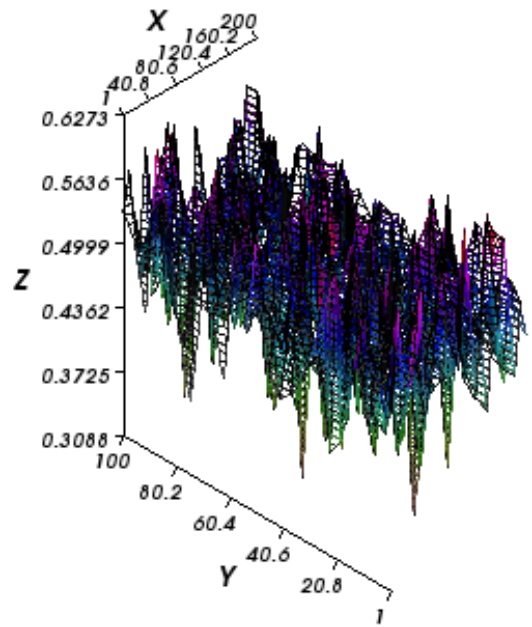
Slice: (:,:) Filter:

Type: REAL*8(100,200)

Field	Value
(1,1)	3.92086819432386e-07
(2,1)	0.0254804427576426
(3,1)	0.352516161261067
(4,1)	0.666914481524251
(5,1)	0.963055531894656
(6,1)	0.838288203465982
(7,1)	0.335355043646496
(8,1)	0.915327203368213
(9,1)	0.795863676652503

sum_buggy::a(1:100,1:200)

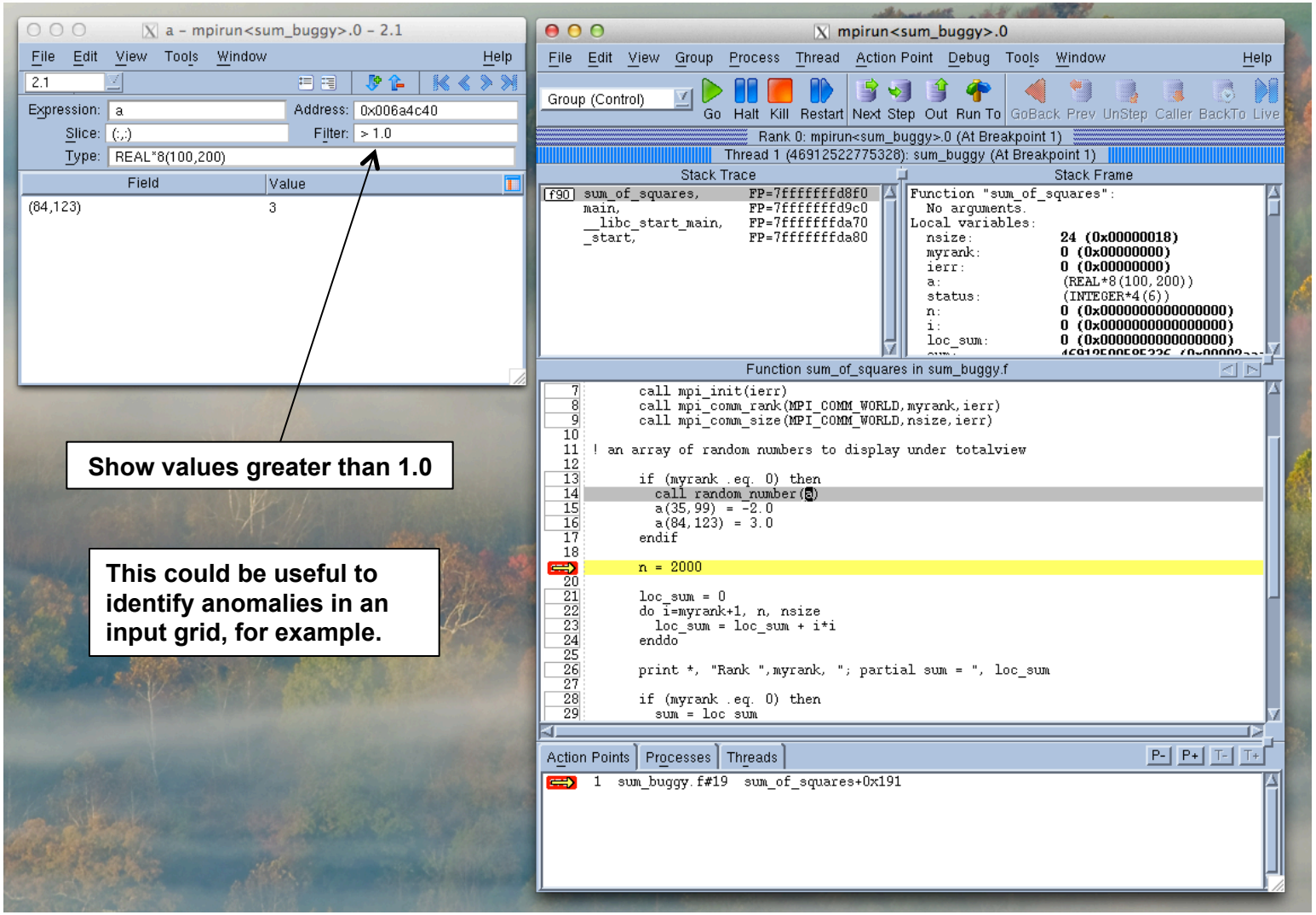
File View Window



Control which slice of the array to display by entering a constant value for one or more dimension.

Question? Use the Webex chat facility to ask the Host

Result of Data Filter

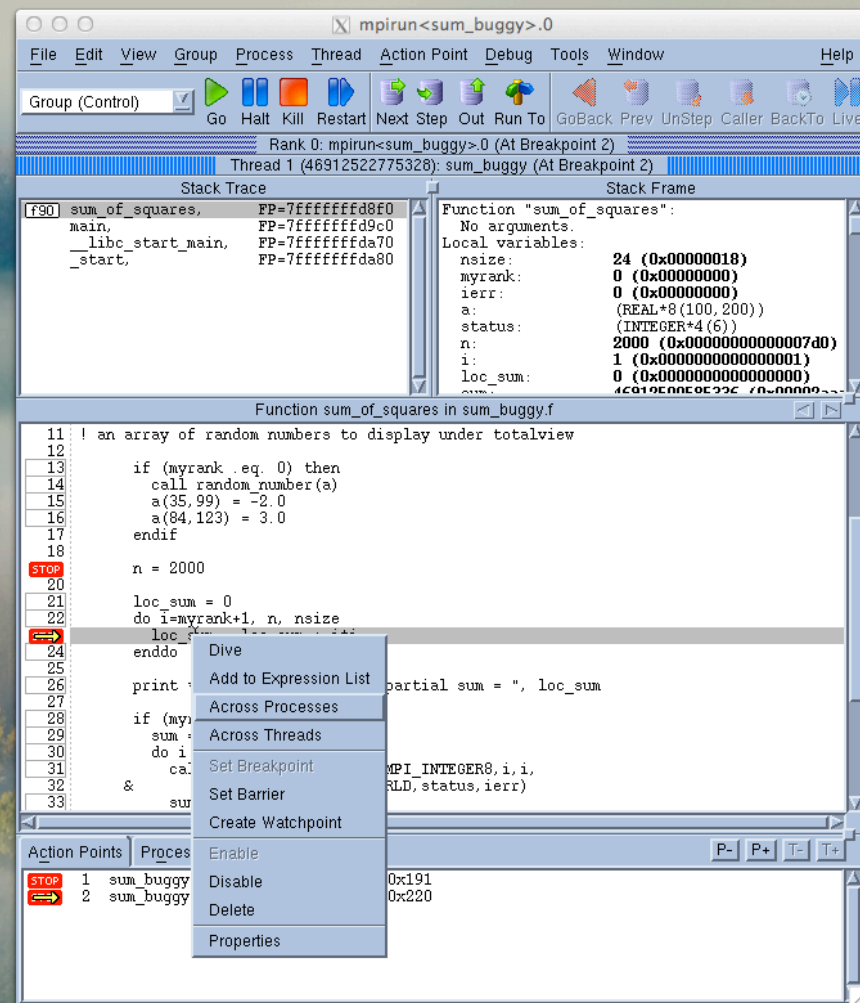


Show values greater than 1.0

This could be useful to identify anomalies in an input grid, for example.

Question? Use the Webex chat facility to ask the Host

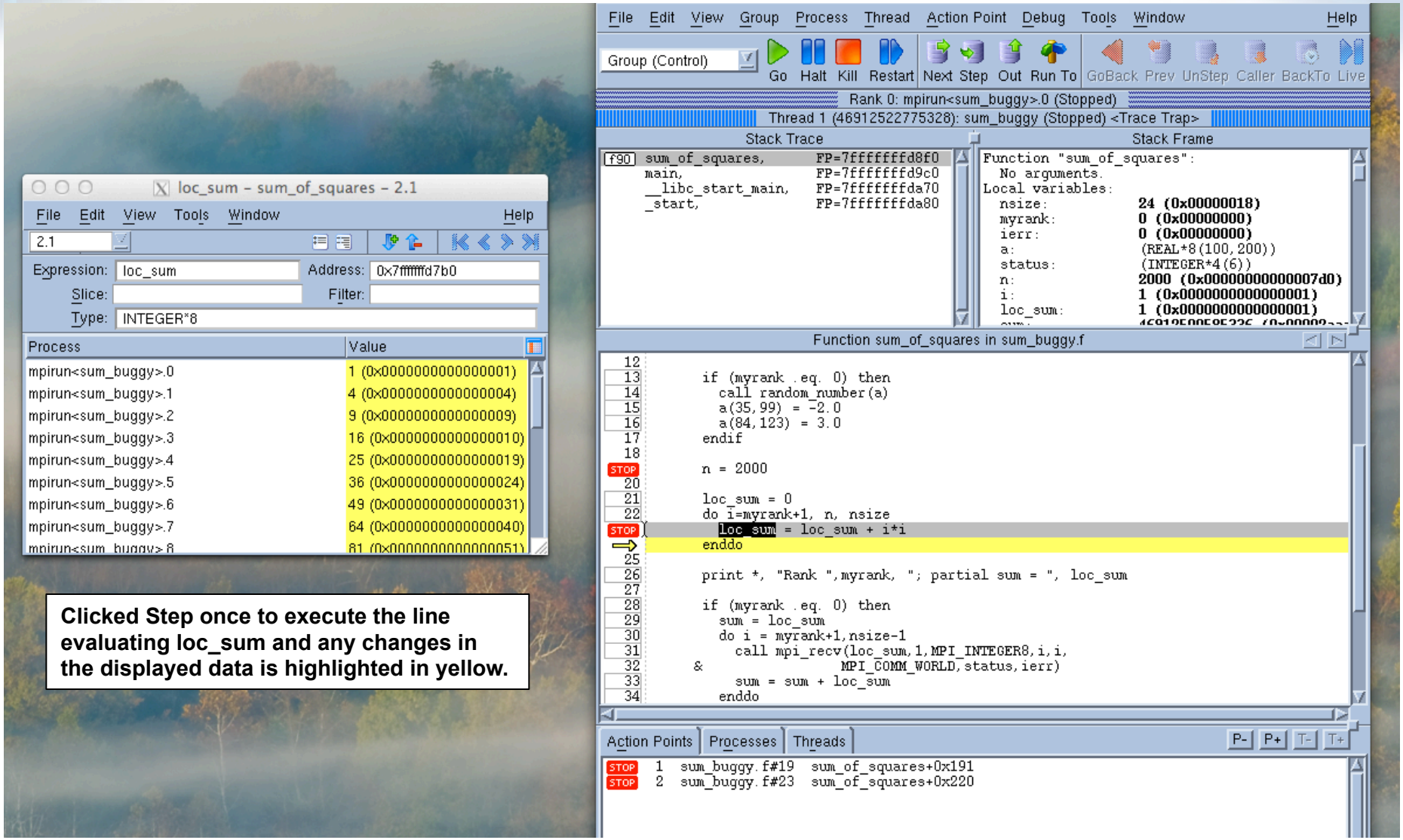
Viewing data across processes



Set a second breakpoint, press "Run To" second breakpoint, and display value of loc_sum across all processes.

Question? Use the Webex chat facility to ask the Host

Values that change are highlighted



Clicked Step once to execute the line evaluating loc_sum and any changes in the displayed data is highlighted in yellow.

Question? Use the Webex chat facility to ask the Host

Debugging program hang

```
xterm
Reading symbols for process 21, executing "/home4/jchang/Su
sum_buggy"
Reading symbols for process 22, executing "/home4/jchang/Su
sum_buggy"
Reading symbols for process 23, executing "/home4/jchang/Su
sum_buggy"
Reading symbols for process 24, executing "/home4/jchang/Su
sum_buggy"
Reading symbols for process 25, executing "/home4/jchang/Su
sum_buggy"
 Rank          0 ; partial sum =          111942516
 Rank         16 ; partial sum =          110607875
 Rank          8 ; partial sum =          109283859
 Rank         15 ; partial sum =          110441792
 Rank          7 ; partial sum =          113119104
 Rank         23 ; partial sum =          111775104
 Rank          6 ; partial sum =          112950516
 Rank         14 ; partial sum =          110275875
 Rank          5 ; partial sum =          112782096
 Rank         22 ; partial sum =          111607859
 Rank          4 ; partial sum =          112613844
 Rank
 Rank                                     760
 Rank                                     780
 Rank                                     120
 Rank                                     367
 Rank                                     344
 Rank                                     120
 Rank                                     367
 Rank                                     539
 Rank                                     780
 Rank         18 ; partial sum =          110940539
 Rank         17 ; partial sum =          110774124
 Rank          1 ; partial sum =          112110096
```

Click on second breakpoint to remove it, and hit "Go" to continue running. The partial sum is printed out by each rank, but then the program just hangs.

Now click on "Halt" to stop the processes and investigate where they are in the program.

```
File Edit View Group Process Thread Action Point Debug Tools Window Help
Group (Control)   Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live
Rank 0: mpirun<sum_buggy>.0 (Running)
Thread 1 (46912522775328): sum_buggy (Running)
Stack Trace                                  Stack Frame
f90 sum_of_squares,     FP=7fffffffd8f0      Thread must be stopped for frame display.
    main,               FP=7fffffffd9c0
    __libc_start_main,  FP=7fffffffda70
    _start,             FP=7fffffffda80

Function sum_of_squares in sum_buggy.f
12
13        if (myrank .eq. 0) then
14          call random_number(a)
15          a(35,99) = -2.0
16          a(84,123) = 3.0
17        endif
18
STOP      n = 2000
20
21        loc_sum = 0
22        do i=myrank+1, n, nsize
23          loc_sum = loc_sum + i*i
          enddo
25
26        print *, "Rank ",myrank, "; partial sum = ", loc_sum
27
28        if (myrank .eq. 0) then
29          sum = loc_sum
30          do i = myrank+1,nsize-1
31            call mpi_recv(loc_sum,1,MPI_INTEGER8,i,i,
32      &                   MPI_COMM_WORLD,status,ierr)
33            sum = sum + loc_sum
34          enddo

Action Points  Processes  Threads   P- P+ T- T+
STOP  1  sum_buggy.f#19  sum_of_squares+0x191
```


Question? Use the Webex chat facility to ask the Host

Stack trace

The Source pane shows the binary code deep inside the MPI library. Click on the last stack of your source code to see where it is calling the MPI routine.

mpirun<sum_buggy>.0

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live

Rank 0: mpirun<sum_buggy>.0 (Stopped)

Thread 1 (46912522775328): sum_buggy (Stopped) <Stop Signal>

Stack Trace

```
C  MPI_SGI_ib_progress_multirail, FP=7ff
C  MPI_SGI_progress,              FP=7fffffffd670
C  MPI_SGI_request_wait,          FP=7fffffffd6a0
C  PMPI_Recv,                     FP=7fffffffd700
   pmpi_recv_,                    FP=7fffffffd770
f90 sum_of_squares,               FP=7fffffffd8f0
   main,                          FP=7fffffffd9c0
   __libc_start_main,             FP=7fffffffda70
   _start,                        FP=7fffffffda80
```

Stack Frame

```
Function "MPI_SGI_ib_progress_multirail":
  No parameters.
Block "$b1":
  cq:              <Bad address: 0x00000000>
Local variables:
  wc:              (struct ibv_wc)
  i:               <Bad address: 0x00000000>
  num_boards:      0x00000001 (1)
  vchan:           0x2aaaab20c23d (&mpi_sgi_s
  ret_cnt:         '\000' (0x00, or 0)
```

Function MPI_SGI_ib_progress_multirail in ibdev_multirail.c

```
        0x2aaaaaf62976:   0x48  shll    $7,%rcx
        0x2aaaaaf62977:   0xc1
        0x2aaaaaf62978:   0xe1
        0x2aaaaaf62979:   0x07
        0x2aaaaaf6297a:   0x48  addl    0(%rbp),%rcx
        0x2aaaaaf6297b:   0x03
        0x2aaaaaf6297c:   0x4d
        0x2aaaaaf6297d:   0x00
3886    0x2aaaaaf6297e:   0x0f  movzbl  16(%rcx),%eax
        0x2aaaaaf6297f:   0xb6
        0x2aaaaaf62980:   0x41
        0x2aaaaaf62981:   0x10
        0x2aaaaaf62982:   0x84  testb   %al,%al
        0x2aaaaaf62983:   0xc0
        0x2aaaaaf62984:   0x0f  jne     0x2aaaaaf62b40
        0x2aaaaaf62985:   0x85
        0x2aaaaaf62986:   0xb6
        0x2aaaaaf62987:   0x01
        0x2aaaaaf62988:   0x00
        0x2aaaaaf62989:   0x00
3881    0x2aaaaaf6298a:   0x41  addl    $1,%r13d
        0x2aaaaaf6298b:   0x83
        0x2aaaaaf6298c:   0xc5
```

Action Points | Processes | Threads P- P+ T- T+

```
STOP  1  sum_buggy.f#19  sum_of_squares+0x191
```


Question? Use the Webex chat facility to ask the Host

View from two snapshots of the Process Window

mpirun<sum_buggy>.0

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live

Rank 0: mpirun<sum_buggy>.0 (Stopped)

Thread 1 (46912522775328): sum_buggy (Stopped) <Stop Signal>

Stack Trace

```
C   MPI_SGI_ib_progress_multirail, FP=7fff
C   MPI_SGI_progress,              FP=7fffffffd670
C   MPI_SGI_request_wait,          FP=7fffffffd6a0
C   PMPI_Recv,                     FP=7fffffffd700
    pmpi_recv_,                    FP=7fffffffd770
f90 sum_of_squares,                FP=7fffffffd8f0
    main,                          FP=7fffffffd9c0
    __libc_start_main,             FP=7fffffffda70
    _start,                        FP=7fffffffda80
```

Stack Frame

```
Function "sum_of_squares":
  No arguments.
Local variables:
  nsize:      24 (0x00000018)
  myrank:     0 (0x00000000)
  ierr:       0 (0x00000000)
  a:          (REAL*8(100,200))
  status:     (INTEGER*4(6))
  n:          2000 (0x00000000000007d0)
  i:          1 (0x0000000000000001)
  loc_sum:    111942516 (0x0000000006ac1
```

Function sum_of_squares in sum_buggy.f

```
STOP      n = 2000
 20
 21       loc_sum = 0
 22       do i=myrank+1, n, nsize
 23         loc_sum = loc_sum + i*i
 24       enddo
 25
 26       print *, "Rank ",myrank, "; partial sum = ", loc_sum
 27
 28       if (myrank .eq. 0) then
 29         sum = loc_sum
 30         do i = myrank+1,nsize-1
 ⇨            call mpi_recv(loc_sum,1,MPI_INTEGER8,i,i,
 32     &                  MPI_COMM_WORLD,status,ierr)
 33           sum = sum + loc_sum
 34         enddo
 35         print *, 'sum of squares is = ',sum
 36         print *, 'correct answer is = ',n*(n+1)*(2*n+1)/6
 37       else
 38         call mpi_send(loc_sum,1,MPI_INTEGER8,0,i,
 39     &                  MPI_COMM_WORLD,ierr)
 40       endif
 41
```

Action Points Processes Threads P- P+ T- T+

STOP 1 sum_buggy.f#19 sum_of_squares+0x191

Rank 0 is at the MPI_recv

mpirun<sum_buggy>.2

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) Go Halt Kill Restart Next Step Out Run To GoBack Prev UnStep Caller BackTo Live

Rank 2: mpirun<sum_buggy>.2 (Stopped)

Thread 1 (46912522775328): sum_buggy (Stopped) <Stop Signal>

Stack Trace

```
    __nanosleep_nocancel,          FP=7fffffffd5b0
C   MPI_SGI_millisleep,            FP=7fffffffd5f0
C   MPI_SGI_slow_request_wait,     FP=7fffffff
C   MPI_SGI_slow_barrier,          FP=7fffffffd6f0
C   MPI_SGI_finalize,              FP=7fffffffd740
C   PMPI_Finalize,                 FP=7fffffffd760
    pmpi_finalize_,                FP=7fffffffd780
f90 sum_of_squares,                FP=7fffffffd8f0
    main,                          FP=7fffffffd9c0
    __libc_start_main,             FP=7fffffffda70
    _start,                        FP=7fffffffda80
```

Stack Frame

```
Function "sum_of_squares":
  No arguments.
Local variables:
  nsize:      24 (0x00000018)
  myrank:     2 (0x00000002)
  ierr:       0 (0x00000000)
  a:          (REAL*8(100,200))
  status:     (INTEGER*4(6))
  n:          2000 (0x00000000000007d0)
  i:          2019 (0x00000000000007e3)
  loc_sum:    112277844 (0x0000000006b13
```

Function sum_of_squares in sum_buggy.f

```
 30         do i = myrank+1,nsize-1
 31           call mpi_recv(loc_sum,1,MPI_INTEGER8,i,i,
 32     &                  MPI_COMM_WORLD,status,ierr)
 33           sum = sum + loc_sum
 34         enddo
 35         print *, 'sum of squares is = ',sum
 36         print *, 'correct answer is = ',n*(n+1)*(2*n+1)/6
 37       else
 38         call mpi_send(loc_sum,1,MPI_INTEGER8,0,i,
 39     &                  MPI_COMM_WORLD,ierr)
 40       endif
 41
 ⇨        call mpi_finalize(ierr)
 43       end
 44
 45
```

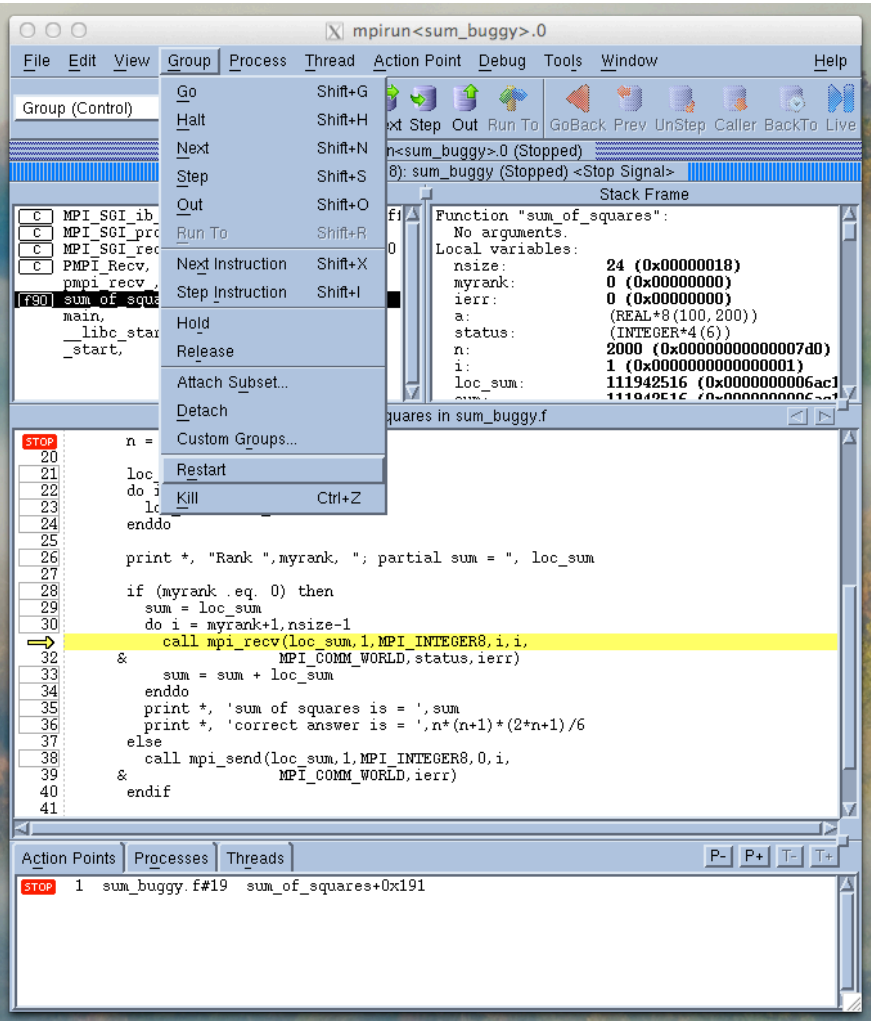
Action Points Processes Threads P- P+ T- T+

STOP 1 sum_buggy.f#19 sum_of_squares+0x191

Rank 1 (and all the other ranks) are at MPI_finalize

Question? Use the Webex chat facility to ask the Host

Re-run the program and set a breakpoint at the MPI_send



Question? Use the Webex chat facility to ask the Host

Bug Revealed

Rank 1: mpirun<sum_buggy>.1 (At Breakpoint 4)

Thread 1 (46912522775328): sum_buggy (At Breakpoint 4)

Stack Trace

```
f90  sum_of_squares,      FP=7fffffffd8f0
     main,                FP=7fffffffd9c0
     __libc_start_main,   FP=7fffffffda70
     _start,              FP=7fffffffda80
```

Stack Frame

```
Function "sum_of_squares":
  No arguments.
Local variables:
  nsize:      24 (0x00000018)
  myrank:     1 (0x00000001)
  ierr:       0 (0x00000000)
  a:          (REAL*8(100,200))
  status:     (INTEGER*4(6))
  n:          2000 (0x00000000000007d0)
  i:          2018 (0x00000000000007e2)
  loc_sum:    112110096 (0x0000000006aea
```

Function sum_of_squares in sum_buggy.f

```
26        print *, "Rank ",myrank, "; partial sum = ", loc_sum
27
28        if (myrank .eq. 0) then
29          sum = loc_sum
30          do i = myrank+1,nsize-1
31            call mpi_recv(loc_sum,1,MPI_INTEGER8,i,i,
32       &                  MPI_COMM_WORLD,status,ierr)
33            sum = sum + loc_sum
34          enddo
35          print *, 'sum of squares is = ',sum
36          print *, 'correct answer is = ',n*(n+1)*(2*n+1)/6
37        else
38          call mpi_send(loc_sum,1,MPI_INTEGER8,0,i,
39       &                  MPI_COMM_WORLD,ierr)
40        endif
41
42        call mpi_finalize(ierr)
43        end
44
45
```

i – sum_of_squares – 51.1

File Edit View Tools Window Help

51.1

Expression: i Address: 0x7fffffffd7d0

Type: INTEGER*8

Value

2018 (0x00000000000007e2)

Mis-match of tags in the send and recv

Action Points | Processes | Threads P- P+ T- T+

```
STOP  1  sum_buggy.f#19   sum_of_squares+0x191
      4  sum_buggy.f#38   sum_of_squares+0x602
```


Question? Use the Webex chat facility to ask the Host

Where to find additional information on using TotalView

Online videos produced by TotalView developers and engineers:

http://www.roguewave.com/products/totalview/resources/videos.aspx

- **Getting Started with TotalView**
- **Debugging MPI**
- **Introducing C++View**
- **Setting Breakpoint References for Threads**
- **Viewing Data Across Threads**
- **Threads Navigation**
- **Asynchronous Thread Control**
- **Debugging OpenMP**
- **C++ STL Type Transformations**
- **Memory Debugging with MemoryScape**
- **Memory Debugging with Red Zones**
- **Deterministic Replay with ReplayEngine**

Question? Use the Webex chat facility to ask the Host

Online Documentation on Pleiades

```
pfe12> which totalview
/nasa/sles11/totalview/toolworks/totalview.8.9.2-1/bin/totalview
pfe12> cd /nasa/sles11/totalview/toolworks/totalview.8.9.2-1/doc
pfe12> ls
pdf/
pfe12> cd pdf
pfe12> pwd
/nasa/sles11/totalview/toolworks/totalview.8.9.2-1/doc/pdf
pfe12> ls
MemoryScape_Installation_Guide.pdf
MemoryScape_New_Features_Guide.pdf
MemoryScape_User_Guide.pdf
ReplayEngine_Getting_Started_Guide.pdf
ReplayEngine_New_Features_Guide.pdf
TotalView_Installation_Guide.pdf
TotalView_New_Features_Guide.pdf
TotalView_Platforms_and_System_Requirements.pdf
TotalView_Reference_Guide.pdf
TotalView_User_Guide.pdf
```


Question? Use the Webex chat facility to ask the Host