

GDB4HPC, COMPARATIVE DEBUGGING

John Vogt : SUPS Development

October 15 , 2026

GDB4HPC INTRODUCTION

John Vogt, SUPS development

October 15. 2026

debugging

[de-buhg-ing] -verb.

1. being the detective in a crime movie where you are also the murderer.



AGENDA

- Demonstrate a spelunking trip through SNAP
- Comparative Debugging
- GPU's, threads, C++ data
- Additional Resources
- Status of gdb4hpc



EXAMPLE WALKTHROUGH

Subtitle here only if needed

START GDB4HPC

```
[jvogt@pea2k src]$ module load gdb4hpc
[jvogt@pea2k src]$ gdb4hpc
gdb4hpc 4.14.1 - Cray Line Mode Parallel
Debugger
With Cray Comparative Debugging
Technology.
Copyright 2007-2022 Hewlett Packard
Enterprise Development LP.
Copyright 1996-2016 University of
Queensland. All Rights Reserved.


Type "help" for a list of commands.
Type "help <cmd>" for detailed help about
a command.
dbg all>
```



GET HELP

```
dbg all> help
assign          Change the value of an application or debugger
variable.
attach          Attach to an application.
backtrace       Print backtrace of all stack frames.
break           Set breakpoint at specified line or function.
build           Build an assertion script.
catch           Set catchpoint on specified event.
checkpoint      Fork program for restoration at a later point.
compare         Compare the contents of two variables.
continue        Continue execution of application.
cuda            Cuda debugging functions.
decomposition   Define a decomposition scheme.
defset          Create a set of processes.
delete          Delete a breakpoint or checkpoint.
disable         Disable a breakpoint.
down            Move down one or more stack frames.
enable          Enable a breakpoint.
finish          Execute application until each rank's current function
returns.
focus           Set the current process set.
frame           Print the currently selected stack frame.
gdbmode         Enter direct gdb mode (experimental).
halt            Halt execution of attached/launched application.
help            Display help information about commands.
info            Display information about the application being
debugged.
kill            Kill attached/launched application.
launch          Launch an application.
list            List source for specified function or line.
maint           Commands for use by debugger maintainers.
next            Step application, proceeding through subroutine calls.
ompd            OpenMP debugging functions.
print           Print the value of an expression.
ptype           Print definition of a type.
quit            Exit the debugger.
release         Release attached/launched application from debugger
control.
restart         Restore program to state of previously created
checkpoint.
run             Release a procset and re-launch it, transferring any
procset-specific settings/breakpoints/etc.
session         Create a new WLM session.
set             Set information about the debugger environment.
show            Show information about the debugger environment.
source          Read debugger commands from a file.
start           Start executing an assertion script.
step            Step application until it reaches a different source
line.
stop            Stop the currently executing assertion script.
tbreak          Set a temporary breakpoint at specified line or
function.
thread          Thread control functions.
unset           Unset information in the debugger environment.
up              Move up one or more stack frames.
usage           Display usage information about script mode commands.
viewset         Display information about process sets.
watch           Set watchpoint on specified expression.
whatis          Print the data type of an expression.
dbg all>
```



LAUNCH THE APPLICATION

```
dbg all> !launch
> launch $a{16} ./csnap_db --launcher-args="-n 16 -N 8 -c 2" -a "../inp162 tmp.out"
Starting application, please wait...
Creating MRNet communication network...
Waiting for debug servers to attach to MRNet communications network...
Timeout in 400 seconds. Please wait for the attach to complete.
Number of dbgsrvs connected: [1];  Timeout Counter: [0]
Number of dbgsrvs connected: [1];  Timeout Counter: [1]
Number of dbgsrvs connected: [16];  Timeout Counter: [0]
Finalizing setup...
Launch complete.
a{0..15}: Initial breakpoint, in main
dbg all>
```

- Using shell history '!' to re-enter the last launch command I used
- We are launching 16 ranks
- $a{16} declared the procset – an array we can use to reference a rank or set of ranks:
  - E.g $a{1}, or $a{1,4} or $a{0..4,8} represent 1, 2 and 6 ranks respectively.
- It show the progress setting up the full application and leaves it stopped in main



SHOW WHERE WE ARE

```
dbg all> frame
a{0..15}: #0  0x000000000044675d in inner_module::inner at /home/users/jvogt/coe-snap-runs/src/inner.f90:45
dbg all> list
a{0..15}: 45     SUBROUTINE inner ( inno, iits, t, do_grp, ng_per_thrd, nnstd_used,   &
a{0..15}: 46       grp_act )
a{0..15}: 47
a{0..15}: 48   !---------------------------------------------------------------------
a{0..15}: 49   !
a{0..15}: 50   ! Do a single inner iteration for all groups. Calculate the total source
a{0..15}: 51   ! for each group and sweep the mesh over octants. Repeat for all groups
a{0..15}: 52   ! unless the group is converged according to inrdone(g).
a{0..15}: 53   !
a{0..15}: 54   !---------------------------------------------------------------------
dbg all>
```

- Calling list again will print the next 10 lines



SET A BREAKPOINT AND RUN TO A IT

```
dbg all> b inner_module::inner
a{0..15}: Breakpoint 1: file /home/users/jvogt/coe-snap-
runs/src/inner.f90, line 45.
dbg all> c
<$a>:           keyword Iteration Monitor
<$a>:
**********************************************************
**********************
<$a>:
<$a>:  ****************************
<$a>:   Time Cycle   1
<$a>:   Outer
a{0..15}: Breakpoint 1, inner_module::inner at
/home/users/jvogt/coe-snap-runs/src/inner.f90:45
dbg all>
```

- After “c” (or ”cont” or “continue”) the programe runs freely. Gdb4print out the message once the program is stopped at our breakpoint.
- The ranks are running independently so they may not all hit the breakpoint at the same time.
- Instead of a{0..15}, we might have seen something like a{1..5,7} first, if those 6 ranks had run enough faster.



STEP THROUGH THE CODE

```
dbg all> n
a{0..15}: Breakpoint 1, inner_module::inner at
/home/users/jvogt/coe-snap-runs/src/inner.f90:45
dbg all> n
a{0..15}: 46     grp_act )
dbg all> n
a{0..15}: 79   !$OMP MASTER
dbg all> n
a{0..15}: 81     CALL wtime ( t1 )
dbg all> n
a{0..15}: 83     do_grp = 1
dbg all> p do_grp
a{0..15}: (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1)
dbg all> n
a{0..15}: 84     WHERE( inrdone ) do_grp = 0
dbg all> n
a{0..15}: 86     CALL assign_thrd_set ( do_grp, ng,
ng_per_thrd, ny*nz, nnstd_used, &
dbg all>
```



LET IT RUN FOR AWHILE

```
dbg all> c -a
a{0..15}: 90   !$OMP BARRIER
dbg all> halt -a
a{0..3,5..9,11..15}: Halt could not report a location
a{4}: Application halted in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{10}: Application halted in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:174
dbg all> frame
a{0}: #0  0x0000000000460d01 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:312
a{1}: #0  0x0000000000463a6a in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:399
a{2}: #0  0x000000000045dc6d in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:177
a{3}: #0  0x0000000000463f5d in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{4}: #0  0x000000000046415e in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{5}: #0  0x0000000000462946 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:359
a{6}: #0  0x0000000000463eff in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{7}: #0  0x0000000000463f2e in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{8}: #0  0x0000000000464019 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{9}: #0  0x000000000045eeb2 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:256
a{10}: #0  0x000000000045da1c in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:174
a{11}: #0  0x0000000000463edf in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
```

- Let's just let it run for a while to hopefully get to someplace interesting
- I let it run for like 10 seconds after continue
- The –a refers to starting and stopping *every* thread.
- Since the ranks run independently, we end up at different locations.

- (note to self: why is frame printing the frame address – it kept it from aggregating the ranks at line 417)

TAKE A CLOSER LOOK AT ONE RANK

```
dbg all> focus $a{11}
dbg a_temp> list
a{11}: 417          fluxm(:,i,j,k) = fluxm(:,i,j,k) +
ec(m,2:cmom)*psi(m)
a{11}: 418        END DO
a{11}:
419  !---------------------------------------------------------------
a{11}: 420  !
a{11}: 421  !   Finish the loops
a{11}:
422  !---------------------------------------------------------------
a{11}: 423
a{11}: 424      END DO ic_loop
a{11}: 425      END DO j_loop
a{11}: 426      END DO k_loop
dbg a_temp> p i
a{11}: 143
dbg a_temp> p j
a{11}: 1
dbg a_temp> n
a{11}: Program received signal 0.
a{11}: In dim3_sweep_module::dim3_sweep at /home/users/jvogt/coe-snap-
runs/src/dim3_sweep.f90:417
dbg a_temp> n
a{11}: dim3_sweep_module::dim3_sweep at /home/users/jvogt/coe-snap-
runs/src/dim3_sweep.f90:418
a{11}: 418       END DO
dbg a_temp> n
a{11}: 415         psi(m) = w(m)*psi(m)
dbg a_temp>
```

- After focus $a{11} every command affects only rank 11.
- In this case, the other ranks are all stopped, but that's not required. Any or all of them could be running.
- Though in practice, they might end up stopped at an MPI construct waiting for a message from rank 11.



PRINT SOME MORE VALUES

```
dbg a_temp> p m
a{11}: 14
dbg a_temp> p nang
a{11}: 48
dbg a_temp> n
a{11}: 416          flux0(i,j,k) = flux0(i,j,k) + psi(m)
dbg a_temp> p i
a{11}: 143
dbg a_temp> p j
a{11}: 1
dbg a_temp> p k
a{11}: 4
dbg a_temp> whatis psi
a{11}: REAL(8) (48)
dbg a_temp> whatis flux0
a{11}: REAL(8) (640,4,4)
dbg a_temp> p flux(i,j,k)
a{11}: No symbol "flux(i,j,k)" in current context
dbg a_temp> p flux0(i,j,k)
a{11}: 2.36447164782482e-15
dbg a_temp> p psi(m)
a{11}: 1.43406229584288e-15
dbg a_temp> n
a{11}: 417          fluxm(:,i,j,k) = fluxm(:,i,j,k) +
ec(m,2:cmom)*psi(m)
dbg a_temp> p flux0(i,j,k)
a{11}: 2.36447164782482e-15
dbg a_temp>
```

- Note “whatis” to get the type of an expression.

ARRAY SLICES AND THE “” ESCAPE FOR DIRECT GDB ACCESS

```
dbg a_temp> p fluxm(:,i,j,k)
syntax error, unexpected COLON, expecting RPAREN or
COMMA
dbg a_temp> p "fluxm(:,i,j,k)"
a{11}: (-6.4517535400996e-16, 2.95337943698905e-16,
-1.35195638106266e-16, -4.82518503535147e-17,
2.20844727178114e-17, -1.01079239556471e-17,
4.62635301687405e-18, -2.11747087659864e-18, -
3.62927733412679e-18, 1.6608419632207e-18, -
7.60043115834438e-19, 3.4781629183552e-19, -
1.5917079489087e-19, 7.28414573359406e-20, -
3.333463058281e-20)
dbg a_temp> whatis fluxm
a{11}: REAL(8) (15,640,4,4)
dbg a_temp> p fluxm(1..15,i,j,k)
a{11}: (-6.4517535400996e-16, 2.95337943698905e-16,
-1.35195638106266e-16, -4.82518503535147e-17,
2.20844727178114e-17, -1.01079239556471e-17,
4.62635301687405e-18, -2.11747087659864e-18, -
3.62927733412679e-18, 1.6608419632207e-18, -
7.60043115834438e-19, 3.4781629183552e-19, -
1.5917079489087e-19, 7.28414573359406e-20, -
3.333463058281e-20)
dbg a_temp>
```

- Gdb4hpc does not understand ’:’ for an array range.
- But putting an expression in double quotes bypassed gdb4hpc’s parser
- Gdb4hpc does provide the ”..” operator to get a range, which also works in C/C++.

- Note to self: teach the gdb4hpc parser about ‘:’.



MORE NAVIGATION

```
dbg a_temp> b 432
a{11}: Breakpoint 2: file /home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90, line 432.
dbg a_temp> c
a{11}: 416         flux0(i,j,k) = flux0(i,j,k) + psi(m)
dbg a_temp> p multiswp
a{11}: 1
dbg a_temp> n
a{11}: Breakpoint 2, dim3_sweep_module::dim3_sweep at /home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:432
dbg a_temp> list
a{11}: 437    END SUBROUTINE dim3_sweep
a{11}: 438
a{11}: 439
a{11}: 440  END MODULE dim3_sweep_module
dbg a_temp> finish
a{11}: 437   END SUBROUTINE dim3_sweep
dbg a_temp> finish
a{11}: octsweep_module::octsweep at /home/users/jvogt/coe-snap-runs/src/octsweep.f90:142
a{11}: 142           CALL dim3_sweep ( ich, id, d1, d2, d3, d4, jd, kd, oct, g,   &
dbg a_temp> n
a{11}: sweep_module::sweep at /home/users/jvogt/coe-snap-runs/src/sweep.f90:256
a{11}: 256         CALL octsweep ( g, jd, kd, 0, dummy, 2, ic_op(cor), cor, 0 )
```

- The “finish” commands runs until the exit from the current function

BACKTRACE

```
dbg a_temp> bt
a{11}: #9  snap_main at /home/users/jvogt/coe-snap-
runs/src/snap_main.f90:200
a{11}: #8  translv at /home/users/jvogt/coe-snap-
runs/src/translv.f90:121
a{11}: #7  __kmpc_fork_call at ../../src/kmp_csupport.cpp:358
a{11}: #6  __kmp_fork_call at ../../src/kmp_runtime.cpp:2551
a{11}: #5  __kmp_invoke_task_func at ../../src/kmp_runtime.cpp:7875
a{11}: #4  __kmp_invoke_microtask
a{11}: #3  translv::L_translv__121__par_region0_2_1 at
/home/users/jvogt/coe-snap-runs/src/translv.f90:246
a{11}: #2  outer_module::outer at /home/users/jvogt/coe-snap-
runs/src/outer.f90:129
a{11}: #1  inner_module::inner at /home/users/jvogt/coe-snap-
runs/src/inner.f90:121
a{11}: #0  sweep_module::sweep at /home/users/jvogt/coe-snap-
runs/src/sweep.f90:258
```



NAVIGATE THE CALL STACK

```
dbg a_temp> up
a{11}: #1  0x000000000044744e in inner_module::inner at
/home/users/jvogt/coe-snap-runs/src/inner.f90:121
dbg a_temp> up
a{11}: #2  0x000000000044104e in outer_module::outer at
/home/users/jvogt/coe-snap-runs/src/outer.f90:129
dbg a_temp> info locals
a{11}: Name:iits                    Type:INTEGER(4) (54)
a{11}: Name:otno                    Type:INTEGER(4)
a{11}: Name:t                       Type:INTEGER(4)
a{11}: Name:do_grp                  Type:INTEGER(4) (54)
a{11}: Name:ng_per_thrd             Type:INTEGER(4)
a{11}: Name:nnstd_used              Type:INTEGER(4)
a{11}: Name:grp_act                 Type:INTEGER(4) (54,2)
a{11}: Name:iitm_i                  Type:INTEGER(4)
a{11}: Name:t4                      Type:REAL(8)
a{11}: Name:t3                      Type:REAL(8)
a{11}: Name:t2                      Type:REAL(8)
a{11}: Name:t1                      Type:REAL(8)
a{11}: Name:n                       Type:INTEGER(4)
a{11}: Name:g                       Type:INTEGER(4)
a{11}: Name:inno                    Type:INTEGER(4)
a{11}: Name:dft                     Type:REAL(8) (2)
dbg a_temp> do
a{11}: #1  0x000000000044744e in inner_module::inner at
/home/users/jvogt/coe-snap-runs/src/inner.f90:121
dbg a_temp> do
a{11}: #0  0x000000000044fd88 in sweep_module::sweep at
/home/users/jvogt/coe-snap-runs/src/sweep.f90:258
dbg a_temp> do
a{11}: Bottom (innermost) frame selected; you cannot go down.
dbg a_temp>
```

- Up, Down (Do for short) changes frames
- Up <n> or Do <n> by multiple frames.
- Info locals shows all the variable in a frame
- There was a bug here where “list” wasn’t respecting the current frame, so I had to use ”list filename:number” directly.

GO BACK TO THE FULL PROGRAM

```
dbg a_temp> focus $all
dbg all> frame
a{0}: #0  0x0000000000460d01 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:312
a{1}: #0  0x0000000000463a6a in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:399
a{2}: #0  0x000000000045dc6d in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:177
a{3}: #0  0x0000000000463f5d in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{4}: #0  0x000000000046415e in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{5}: #0  0x0000000000462946 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:359
a{6}: #0  0x0000000000463eff in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{7}: #0  0x0000000000463f2e in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{8}: #0  0x0000000000464019 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:417
a{9}: #0  0x000000000045eeb2 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:256
a{10}: #0  0x000000000045da1c in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:174
a{11}: #0  0x000000000044fd88 in sweep_module::sweep at
/home/users/jvogt/coe-snap-runs/src/sweep.f90:258
a{12}: #0  0x0000000000463cd7 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:415
a{13,15}: #0  0x000000000045dc5d in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:177
a{14}: #0  0x000000000045dcd6 in dim3_sweep_module::dim3_sweep at
/home/users/jvogt/coe-snap-runs/src/dim3_sweep.f90:177
```

- focus $all goes back to working with all ranks
- At this point we could continue or quit to exit gdb4hpc



RESULT

- It does only take a few commands to dig into an application
- I miss the gdb integration with emacs or the gdb TUI
- I did have a text editor in the background rather than relying entirely on ”list”
- In practice, I suspect one would compile sweep_3d optimized
  - At least g++ interoperates –g and –O.



MORE RESULTS

- I did find a couple of gdb4hpc bugs
  - It was handling TYPE(0:1) arrays – needed gdbmode
  - ”list” didn’t always respect the current scope
- One feature enhancement
  - It’d be nice to recognized ”(:,I,j)” without the quotes
- One mystery
  - Extra stop signals, is that intel omp?
- One bit of wisdom
  - “Debugger” is the dude, not the tool.
  - It can show the inner workings, but the person at the keyboard has to know when it’s gone wrong.

Still a debugging tool is the difference between paleontology and zoology



COMPARATIVE DEBUGGING

- A technique that takes advantage of two observations
  - Much of our work isn't entirely new but an attempted improvement over an earlier version
  - Automatically determining a result is *wrong* is a lot harder than detecting that a result is *different*

DEMONSTRATION

- Switching to a simple test case
  - I'm not familiar enough with SNAP to realistically perturb it
  - Still fortran, but using PrgEnv-cray



COMPARATIVE DEBUGGING

Subtitle here only if needed

USING THE SOURCE COMMAND TO SIMPLIFY SET UP

```
[jvogt@pegasus tests]$ gdb4hpc
gdb4hpc 4.14.1 - Cray Line Mode Parallel Debugger
With Cray Comparative Debugging Technology.
Copyright 2007-2022 Hewlett Packard Enterprise
Development LP.
Copyright 1996-2016 University of Queensland. All
Rights Reserved.


Type "help" for a list of commands.
Type "help <cmd>" for detailed help about a
command.
dbg all> maint system cat runme
launch $a{3} --args="0 6"
bd_dt/fortran/ftn_arrays.x
launch $b{3} --args="1 6"
bd_dt/fortran/ftn_arrays.x
b 113
b 140


dbg all>
```

- “Maint system” is about be replaced with “shell” to match gdb.
- And “shell –r” will be added to work on the back ends.
- Note there is a different procset name for each: $a and $b.



LAUNCHING TWO APPLICATIONS

```
dbg all> source runme
Starting application, please wait...
Creating MRNet communication network...
Waiting for debug servers to attach to MRNet communications network...
Timeout in 400 seconds. Please wait for the attach to complete.
Number of dbgsrvs connected: [1];  Timeout Counter: [0]
Number of dbgsrvs connected: [1];  Timeout Counter: [1]
Number of dbgsrvs connected: [3];  Timeout Counter: [0]
Finalizing setup...
Launch complete.
a{0..2}: Initial breakpoint, ftn_arrays at
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90:1
Starting application, please wait...
Creating MRNet communication network...
Waiting for debug servers to attach to MRNet communications network...
Timeout in 400 seconds. Please wait for the attach to complete.
Number of dbgsrvs connected: [1];  Timeout Counter: [0]
Number of dbgsrvs connected: [1];  Timeout Counter: [1]
Number of dbgsrvs connected: [3];  Timeout Counter: [0]
Finalizing setup...
Launch complete.
b{0..2}: Initial breakpoint, ftn_arrays at
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90:1
a{0..2}: Breakpoint 1: file
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90, line 113.
b{0..2}: Breakpoint 1: file
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90, line 113.
a{0..2}: Breakpoint 2: file
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90, line 140.
b{0..2}: Breakpoint 2: file
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90, line 140.
dbg all>
```

DEBUGGING IN TANDEM

```
dbg all> n
a{0..2}: 158 end program ftn_arrays
b{0..2}: 158 end program ftn_arrays
dbg all> n
a{0..2}: 37   call get_command_argument(1, arg_application)
b{0..2}: 37   call get_command_argument(1, arg_application)
dbg all> n
a{0..2}: 1 program ftn_arrays
b{0..2}: 1 program ftn_arrays
dbg all> n
a{0..2}: 38   call get_command_argument(2, arg_exponent)
b{0..2}: 38   call get_command_argument(2, arg_exponent)
dbg all> n
a{0..2}: 1 program ftn_arrays
b{0..2}: 1 program ftn_arrays
dbg all> n
a{0..2}: 40   read (arg_application, '(I10)') application
b{0..2}: 40   read (arg_application, '(I10)') application
dbg all> n
a{0..2}: 41   read (arg_exponent, '(I10)') exponent
b{0..2}: 41   read (arg_exponent, '(I10)') exponent
dbg all> n
a{0..2}: 43   call MPI_Init(ierror)
b{0..2}: 43   call MPI_Init(ierror)
dbg all> p exponent
a{0..2}: 6
b{0..2}: 6
dbg all> p application
a{0..2}: 0
b{0..2}: 1
dbg all>
```

- Each time we execute a debugging command it applies to both application
- Cray fortran does some weird line numbering on entrance to the main program, so ignore the line 1 and 158's.
- Note how print print the value from each application.



PRINT AN ARRAY BEFORE IT'S MODIFIED

```
dbg all> c
b{0..2}: Breakpoint 1, ftn_arrays at /home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90:113
a{0..2}: Breakpoint 1, ftn_arrays at /home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90:113
dbg all> list 113
dbg all> p double_2
a{0..2}: ((3.09999990463257, 4.09999990463257, 5.09999990463257, 6.09999990463257, 7.09999990463257, 8.10000038146973,
9.100000381469973, 10.10000038146697), (5.19999980926514, 6.19999980926514, 7.19999980926514, 8.19999980926514, 9.19999980926514,
10.1999998092651, 11.1999998092651, 12.1999998092651), (7.29999971389771, 8.29999923706055, 9.29999923706055, 10.2999992370605,
11.2999992370605, 12.2999992370605, 13.2999992370605, 14.2999992370605), (9.399999961853027, 10.3999996185303, 11.3999996185303,
12.3999996185303, 13.3999996185303, 14.3999996185303, 15.3999996185303, 16.3999996185303), (11.5, 12.5, 13.5, 14.5, 15.5, 16.5,
17.5, 18.5), (13.5999994277954, 14.5999994277954, 15.5999994277954, 16.5999984741211, 17.5999984741211, 18.5999984741211,
19.5999984741211, 20.5999984741211), (15.6999988555908, 16.6999988555908, 17.6999988555908, 18.6999988555908, 19.6999988555908,
20.6999988555908, 21.6999988555908, 22.6999988555908), (17.7999992370605, 18.7999992370605, 19.7999992370605, 20.7999992370605,
21.7999992370605, 22.7999992370605, 23.7999992370605, 24.7999992370605), (19.8999996185303, 20.8999996185303, 21.8999996185303,
22.8999996185303, 23.8999996185303, 24.8999996185303, 25.8999996185303, 26.8999996185303), (22, 23, 24, 25, 26, 27, 28, 29),
(24.0999984741211, 25.0999984741211, 26.0999984741211, 27.0999984741211, 28.0999984741211, 29.0999984741211, 30.0999984741211,
31.0999984741211), (26.1999988555908, 27.1999988555908, 28.1999988555908, 29.1999988555908, 30.1999988555908, 31.1999988555908,
32.1999969482422, 33.1999969482422), (28.2999992370605, 29.2999992370605, 30.2999992370605, 31.2999992370605, 32.2999992370605,
33.2999992370605, 34.2999992370605, 35.2999992370605), (30.3999977111816, 31.3999977111816, 32.3999977111816, 33.3999977111816,
34.3999977111816, 35.3999977111816, 36.3999977111816, 37.3999977111816), (32.5, 33.5, 34.5, 35.5, 36.5, 37.5, 38.5, 39.5),
(34.5999984741211, 35.5999984741211, 36.5999984741211, 37.5999984741211, 38.5999984741211, 39.5999984741211, 40.5999984741211,
41.5999984741211), (36.6999969482422, 37.6999969482422, 38.6999969482422, 39.6999969482422, 40.6999969482422, 41.6999969482422,
42.6999969482422, 43.6999969482422))
b{0..2}: ((3.09999990463257, 4.09999990463257, 5.09999990463257, 6.09999990463257, 7.09999990463257, 8.10000038146973,
9.100000381469973, 10.10000038146697), (5.19999980926514, 6.19999980926514, 7.19999980926514, 8.19999980926514, 9.19999980926514,
10.1999998092651, 11.1999998092651, 12.1999998092651), (7.29999971389771, 8.29999923706055, 9.29999923706055, 10.2999992370605,
11.2999992370605, 12.2999992370605, 13.2999992370605, 14.2999992370605), (9.399999961853027, 10.3999996185303, 11.3999996185303,
12.3999996185303, 13.3999996185303, 14.3999996185303, 15.3999996185303, 16.3999996185303), (11.5, 12.5, 13.5, 14.5, 15.5, 16.5,
17.5, 18.5), (13.5999994277954, 14.5999994277954, 15.5999994277954, 16.5999984741211, 17.5999984741211, 18.5999984741211,
19.5999984741211, 20.5999984741211), (15.6999988555908, 16.6999988555908, 17.6999988555908, 18.6999988555908, 19.6999988555908,
20.6999988555908, 21.6999988555908, 22.6999988555908), (17.7999992370605, 18.7999992370605, 19.7999992370605, 20.7999992370605,
21.7999992370605, 22.7999992370605, 23.7999992370605, 24.7999992370605), (19.8999996185303, 20.8999996185303, 21.8999996185303,
22.8999996185303, 23.8999996185303, 24.8999996185303, 25.8999996185303, 26.8999996185303), (22, 23, 24, 25, 26, 27, 28, 29),
(24.0999984741211, 25.0999984741211, 26.0999984741211, 27.0999984741211, 28.0999984741211, 29.0999984741211, 30.0999984741211,
31.0999984741211), (26.1999988555908, 27.1999988555908, 28.1999988555908, 29.1999988555908, 30.1999988555908, 31.1999988555908,
32.1999969482422, 33.1999969482422), (28.2999992370605, 29.2999992370605, 30.2999992370605, 31.2999992370605, 32.2999992370605,
33.2999992370605, 34.2999992370605, 35.2999992370605), (30.3999977111816, 31.3999977111816, 32.3999977111816, 33.3999977111816,
34.3999977111816, 35.3999977111816, 36.3999977111816, 37.3999977111816), (32.5, 33.5, 34.5, 35.5, 36.5, 37.5, 38.5, 39.5),
(34.5999984741211, 35.5999984741211, 36.5999984741211, 37.5999984741211, 38.5999984741211, 39.5999984741211, 40.5999984741211,
41.5999984741211), (36.6999969482422, 37.6999969482422, 38.6999969482422, 39.6999969482422, 40.6999969482422, 41.6999969482422,
42.6999969482422, 43.6999969482422))
dbg all>
```



COMPARE THE ARRAY CONTENTS

```
dbg all> compare $a::double_2 = $b::double_2
{0..2}: Comparison is true.
```

- Beats examining manually
- !=, <, <=, >, >= also available
- And setting an epsilon



CONTINUE TO WHERE THE ARRAYS NO LONGER MATCH

```
dbg all> c
a{0..2}: Breakpoint 2, ftn_arrays at
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/
ftn_arrays.f90:140
b{0..2}: Breakpoint 2, ftn_arrays at
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/
ftn_arrays.f90:140
dbg all> compare $a::double_2 = $b::double_2
{0..2}: Comparison is false.
The difference delta is:
{0..2}: (1)(1):96.9
dbg all>
```

- I think I've fixed the formatting of the fortran array index on the trunk..
- Currently this just shows up the first 100 differences, but we'll add better controls in the future.



PERFORMING THE COMPARISON VIA AN ASSERTION SCRIPT

```
dbg all> source runme
Shutting down debugger and resuming application for 'a'.
Starting application, please wait...
Creating MRNet communication network...
Waiting for debug servers to attach to MRNet communications
network...
Timeout in 400 seconds. Please wait for the attach to complete.
Number of dbgsrvs connected: [1];  Timeout Counter: [0]
Number of dbgsrvs connected: [1];  Timeout Counter: [1]
Number of dbgsrvs connected: [3];  Timeout Counter: [0]
Finalizing setup...
Launch complete.
a{0..2}: Initial breakpoint, ftn_arrays at
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/
ftn_arrays.f90:1
Shutting down debugger and resuming application for 'b'.
Starting application, please wait...
Creating MRNet communication network...
Waiting for debug servers to attach to MRNet communications
network...
Timeout in 400 seconds. Please wait for the attach to complete.
Number of dbgsrvs connected: [1];  Timeout Counter: [0]
Number of dbgsrvs connected: [1];  Timeout Counter: [1]
Number of dbgsrvs connected: [3];  Timeout Counter: [0]
Finalizing setup...
Launch complete.
```

```
b{0..2}: Initial breakpoint, ftn_arrays at
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fort
ran/ftn_arrays.f90:1
a{0..2}: Breakpoint 1: file
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fort
ran/ftn_arrays.f90, line 113.
b{0..2}: Breakpoint 1: file
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fort
ran/ftn_arrays.f90, line 113.
a{0..2}: Breakpoint 2: file
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fort
ran/ftn_arrays.f90, line 140.
b{0..2}: Breakpoint 2: file
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fort
ran/ftn_arrays.f90, line 140.
dbg all> del
Delete all breakpoints in focus? (y or n) y


dbg all>
```

- Starting over
- Delete the breakpoints; the assertion script will take over stopping and starting.



BUILD THE ASSERTION SCRIPT

```
dbg all> build $sc
> set halt on
> help assert

Summary: Assertion script assert command.
Usage: assert <expr1> = <expr2>
Example command:
assert $app1::var1@"app1.c":10 = $app2::var2@"app2.c":20

Arguments: expr1 (assert expr syntax), expr2 (assert expr syntax)

Assert expression syntax

Expr Usage: <procset>::<var>@"<source file>":<line>
Expr arguments: procset (defined process set name), var (application
                variable), source file (quoted source file name), line (line
                in source file to reference var)
Expr example: $my_app::my_var@"my_source.c":10
<skipping lines here>
> assert $a::double_2@"ftn_arrays.f90":140 = $b::double_2@"ftn_arrays.f90":140
> assert $a::double_2@"ftn_arrays.f90":113 = $b::double_2@"ftn_arrays.f90":113
> end
Assertion script $sc compiled.
dbg all>
```

- The “build” command start defining the assertion script
- “help” gives info related to build (I reserve the right to forget how to use my own application)
- Set halt on tells to stop the application when an assertion fails
- A script can contain multiple assertions
- Each expression includes where to evaluate it.
- Otherwise, it has the same properties as compare – so this one will do a rank by rank comparison



RUN THE SCRIPT

```
dbg all> start $sc
Sending continue to application...
Assertion script $sc finished.
Use "info script $sc results" to view results.
Sending halt to all involved ranks.
a{0..2}: Application halted in ftn_arrays at
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90:140
b{0..2}: Application halted in ftn_arrays at
/home/users/jvogt/rpm_install/gdb4hpc/4.14.1.0/tests/bd_dt/fortran/ftn_arrays.f90:140
Assertion script $sc stopped.


dbg all> info script $sc results
Results summary for script $sc, configured with halt:


Total assertion evaluations: 2
Passed: 1, Passed with warning: 0, Failed: 1, Indeterminate: 0
Assertion results:


Assertion 1:
$a::double_2@"ftn_arrays.f90":140 == $b::double_2@"ftn_arrays.f90":140
Passes: 0, Passes with warnings: 0, Failures: 1.


Assertion 2:
$a::double_2@"ftn_arrays.f90":113 == $b::double_2@"ftn_arrays.f90":113
Passes: 1, Passes with warnings: 0, Failures: 0.
```

- We “start” the script and gdb4hpc takes over control of the session.
- Since we ”set halt on” the script halts the application when assertion 2 fails.

SHOW THE SCRIPT RESULTS

```
dbg all> info script $sc results
Results summary for script $sc, configured with halt:


Total assertion evaluations: 2
Passed: 1, Passed with warning: 0, Failed: 1, Indeterminate: 0
Assertion results:


Assertion 1:
$a::double_2@"ftn_arrays.f90":140 == $b::double_2@"ftn_arrays.f90":140
Passes: 0, Passes with warnings: 0, Failures: 1.


Assertion 2:
$a::double_2@"ftn_arrays.f90":113 == $b::double_2@"ftn_arrays.f90":113
Passes: 1, Passes with warnings: 0, Failures: 0.


dbg all>
```

- The assertion tracks the number of passed and failed comparisons for each assertion
- In a real debugging situation, the assertions would likely be applied to something that gets executed (a lot) more than once.
- You may choose to simply audit the result of any or all of the assertions in the script. That is, not halt when the comparison fails.



WHAT YOU SAW

- Running two applications side by side
- Using the compare operator for one-off checks
- Assertion scripts to automate the checking

NOT SHOWN

- Comparisons aren't light-weight, you'll want to use a refinement strategy to narrow in on the right iteration
- This isn't the only way leverage comparative debugging
  - I've even created temporary versions that computed a result using *both* the old and new versions of a function
- Re-entrant code can be a real friend.
  - Though you'll need gdbmode for the "jump" command.



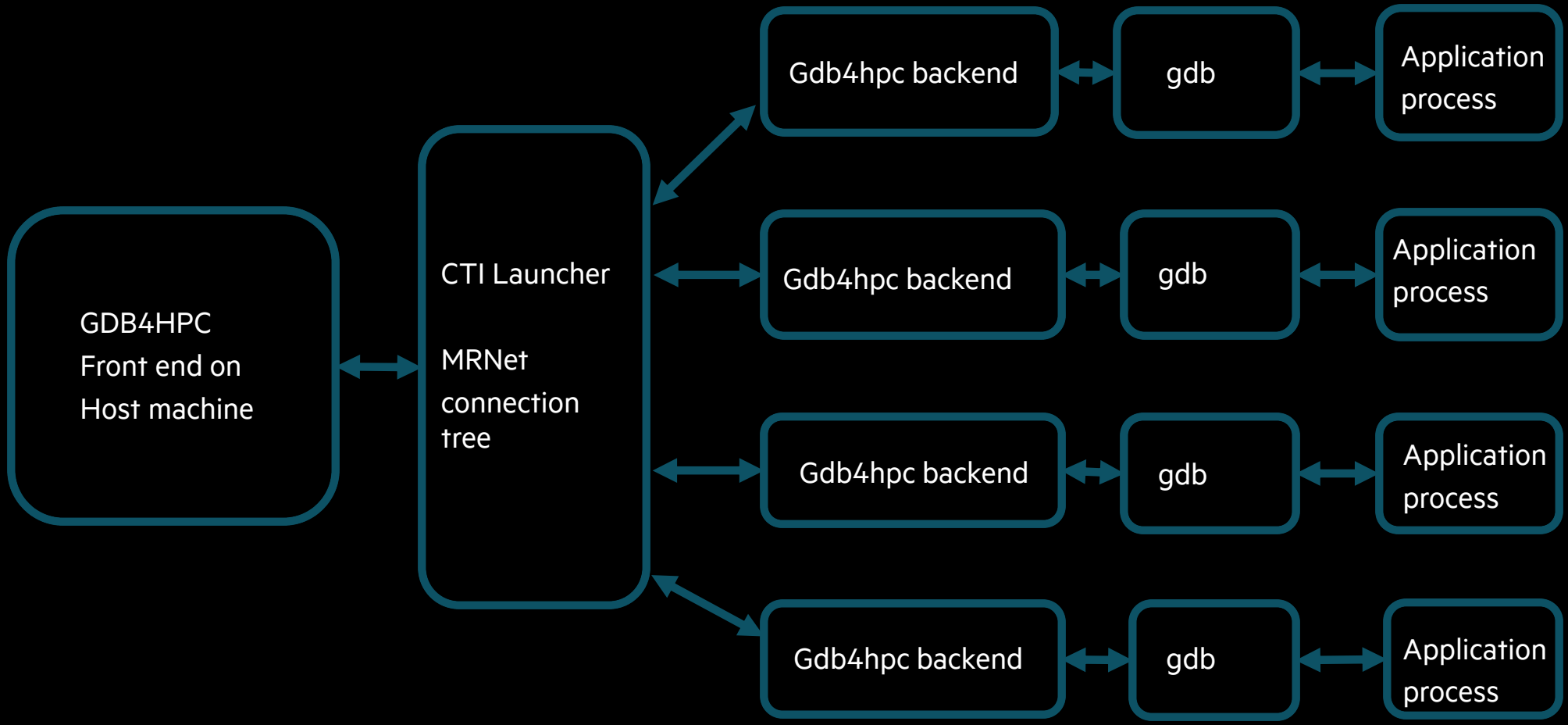
CHANGES SINCE 2020

- Much better C++ support
  - Containers, templated types, smart pointers
  - Aiming for kokkos and raja
- Improved handling for large data arrays
- Improved assertion script handling
  - Easier to use comparisons
  - Better error reporting
  - More readable difference output
- Added a rerun option to launch and keep breakpoints.
- Support for debugging omp and AMD gpus
  - Access to cuda specific functionality
- History is fixed – up and down arrow
- Just added tutorial and “Getting Started” guide
- Eating our own dog food – debugging gdb4hpc *with* gdb4hpc

AGGREGATION, THREADS, GPU, OTHER GDB4HPC TOPICS

Subtitle here only if needed

GDB4HPC ARCHITECTURE



GPU DEBUGGING

- --gpu argument to launch and attach
- Or --gdb=rocgdb
- The program needs to be compiled to support gpu debugging (e.g. –heterogenous-debugging)
- Will try to pick either rocgdb or cuda-gdb based on the application
- Recognizes a “cuda” command – typically for thread focus
- Typically used when focused on a single rank or small group of ranks



AGGREGATION

- The first go to strategy used everywhere in gdbhpc
- Put identical results together
- Both in the UI and in the data handling starting with the MRNet tree
- Mileage may vary..
- Useful far more often than it’s not



GDB4HPC THREAD AGGREGATION

- Based on two observations
  - Modern programs can have way more threads than a user would care to scan through
  - A lot of the threads tend to be doing the same thing

```
dbg all> info threads
Threads for $a:
  Id    Frame
* 1     "cpp_threads.x" main (argc=1, argv=) at bd_dt/C++/cpp_threads.cpp:80
  2-11  "cpp_threads.x" nanosleep () from /lib64/libpthread.so.0
```

- Aggregation does rely on scrubbing out a lot of frame information (pointers)
  - info thread –v is available
- Trivial successful use case
  - Running on a gpu with 4K threads
  - Just find the thread that crashed



THREAD HANDLING EXTENSIONS IN GDB4HPC

- Gdb does have some thread set ability
  - thread apply 2 4 7 9 p j
- Gdb4hpc adds:
  - Ranges : 1-10 or 1..10
  - Sets of ranges: 1,3-10,24,50..70
  - Regular expressions: “worker 1\.[0-9]”
  - Ranges plus regular expressions, etc.
- Extended set form is available for all contexts
  - Info threads 1 4 10-12



NON-STOP MODE

- Original gdb thread model is full stop mode
  - With the “thread” command to focus on one thread at a time
  - Worked pretty well when a program might have a dozen or so threads
  - And not a lot semaphores or mutexes
- Gdb4hpc encountered the same problem in the context of processes
  - Stops the set of processes that are in focus
  - Pull over the car that’s weaving instead of closing the whole freeway
  - It can still stop every rank
    - No illusion of synchronicity
- Gdb does not define a non-stop mode. And scheduler locking.
- Remember to use “c –a” to restart all threads
- Gdb doesn’t automatically focus on the stopped thread
- You can set to full stop mode, but only before the launch (might not be supported in rocgdb for thread debugging)



POINTERS

- Foundational for C/C++
- Defeat aggregation
- Have little meaning in a parallel context
- Gdb4hpc steals a page from java and python

```
dbg all> p ll
a{0..99}: (LinkedList *) [1]
dbg all> p *ll
a{0..99}: {[1] = {val = 0, next = (LinkedList *) [2]}, [2] = {val = 0, next = (LinkedList *) [3]}, [3] = {val = 2,
next = (LinkedList *) [4]}, [4] = {val = 4, next = (LinkedList *) [5]}, [5] = {val = 1, next = (LinkedList *) [6]},
[6] = {val = 3, next = (LinkedList *) [7]}, [7] = {val = 0, next = (LinkedList *) [8]}, [8] = {val = 2, next =
(LinkedList *) [9]}, [9] = {val = 4, next = (LinkedList *) [10]}, [10] = {val = 1, next = (LinkedList *) [11]},
[11] = {val = 3, next = (LinkedList *) [1]}}
dbg all>
```



LOOKING AT POINTERS

- You can still see the pointer values if you need to with a –v option
- Or the gdbmode

```
dbg all> focus $a{0..5}
dbg a_temp> gdbmode
Entering gdb pass-thru mode. Type "end" to exit mode...
    ... gdb banner ...
> p ll
a{1..3,5}:
$2 = (LinkedList *) 0x6e27a0
a{4}:
$2 = (LinkedList *) 0x6e27f0
a{0}:
$2 = (LinkedList *) 0x813bf0
>
```



PARALLEL VALUES

```
dbg all> list 67
a{0..99}: 62 std::vector<float> vf;
a{0..99}: 63 std::vector<double> vd;
a{0..99}: 64 std::vector<int *> ip;
a{0..99}: 65 std::vector<Rectangle > vv;
a{0..99}: 66     LinkedList* ll = new LinkedList;  std::string ss{"So this grasshopper walks into a
bar.."};
a{0..99}: 67 while(i < 10 ){
a{0..99}: 68 vv.push_back(rect);
a{0..99}: 69 ip.push_back((int*)0x1234);
a{0..99}: 70 vd.push_back(i+.5);
a{0..99}: 71 vf.push_back(i+0.5);
dbg all> b 67
a{0..99}: Breakpoint 1: file /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp, line 67.
dbg all> c
a{0..99}: Breakpoint 1, main(int, char**) at
/lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:67
dbg all> p i
a{0..99}: 0
```

```
  ... manipulating the program to make ranks 30-39 different ...
dbg all> p i
a{0..29,40..99}: 2
a{30..39}: 3
dbg all>
```



USING THE PROCESS SET WITH THE SCOPE OPERATOR

- A procset can be used with the :: operator in expressions

```
dbg all> list 62,62
a{0..99}: 62 std::vector<float> vf;
dbg all> b 85
a{0..99}: Breakpoint 3: file /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp, line 85.
dbg all> info br
Procset Num Type          Disp Enb What
a{0..99}: 2   breakpoint    keep y   in main(int, char**) at /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:68
a{0..99}: 3   breakpoint    keep y   in main(int, char**) at /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:85
dbg all> del 2
dbg all> c
a{0..99}: Breakpoint 3, main(int, char**) at /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:85
dbg all> p vf
a{0..99}: [0.5,1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5,9.5]
dbg all> p $a{0}::vf
a{0}: [0.5,1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5,9.5]
```



DEFINING CUSTOM SETS

```
dbg all> defset $odds $a{1,3,5,7,9,11}
dbg all> p $odds::i
a{1,3,5,7,9,11}: 10
dbg all> focus $odds
dbg odds> b f
a{1,3,5,7,9,11}: Breakpoint 4: file /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp, line 105.
dbg odds> c
a{1,3,5,7,9,11}: Breakpoint 4, f(Struct1 const&, Classy const&) at /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:105
dbg odds> bt
a{1,3,5,7,9,11}: #1  main at /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:95
a{1,3,5,7,9,11}: #0  f at /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:105
dbg odds> focus $all
dbg all> info bre
Procset Num Type          Disp Enb What
a{0..99}: 3   breakpoint     keep y   in main(int, char**) at /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:85
a{1,3,5,7,9,11}: 4   breakpoint     keep y   in f(Struct1 const&, Classy const&) at /lus/scratch/jvogt/tests/gdb4hpc/bd_dt/C++/cpp_ds_test1.cpp:105
dbg all>
```

- Motivation

Convenience for programs where not all the ranks are performing the same function



ARRAY DECOMPOSITIONS

- Allows data spread across multiple ranks to be treated as single array
- Supports several forms of data striping and dimension reordering
- Powerful, but non-trivial to use

```
decomposition $block1_3
dimension 120
distribute block
proc_grid 3
end

decomposition $block1_4
dimension 120
distribute block
proc_grid 4
End

decomposition $block2_3
dimension 10,12
distribute *,block
proc_grid *,3
end
```

```
decomposition $block1_3 120 / 3
decomposition $block2_3 10 12/3
```



THE DECOMPOSITION ADAPTER

- For this version $a splits a 120 element array across 3 ranks - 40 elements per rank
- $b splits the (almost) same 120 element array across 4 ranks – 30 elements per rank

```
dbg all> p $block1_3{$a::block1}
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119)
dbg all> p $block1_4{$b::block1}
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, -100, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, -
100, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, -100, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, -100, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118,
119)
dbg all>
```

- The general form is $decomposition{expression}
- Here expression happens to also use the procset scope operator



OTHER FAVORITES

- Attach to a running job. attach $a jobid.stepid
- Source [script]
- Shell / shell –r e.g. shell –r hostname
- Pipe. E.g. pipe p/1 my_array | grep 0x0
- Print formats /x /o /b /1 /csv
- Breakpoint counters / ignore
- Conditional breakpoints
- Gdb/gdbmode : gdb jump 200
- Kokkos/raja support
- Python debugging
- Mpi queue debugging (mpich)
- VSCode plugin*



RESOURCES AND STATUS

Subtitle here only if needed

FURTHER RESOURCES

- https://cpe.ext.hpe.com/docs/latest/debugging-tools/gdb4hpc/guides/getting-started.html
- You can google gdb4hpc, but you'll probably see an older version
- AI tends to know about gdb4hpc



STATUS

- 26.03 is the last scheduled release of CPE and the gdb4hpc bundling
- Under support for 15 months
- Gdb4hpc is now* available as open-source https://github.com/hpe
- * (if not today, then within a week)




THANK YOU
john.leroy.vogt@hpe.com