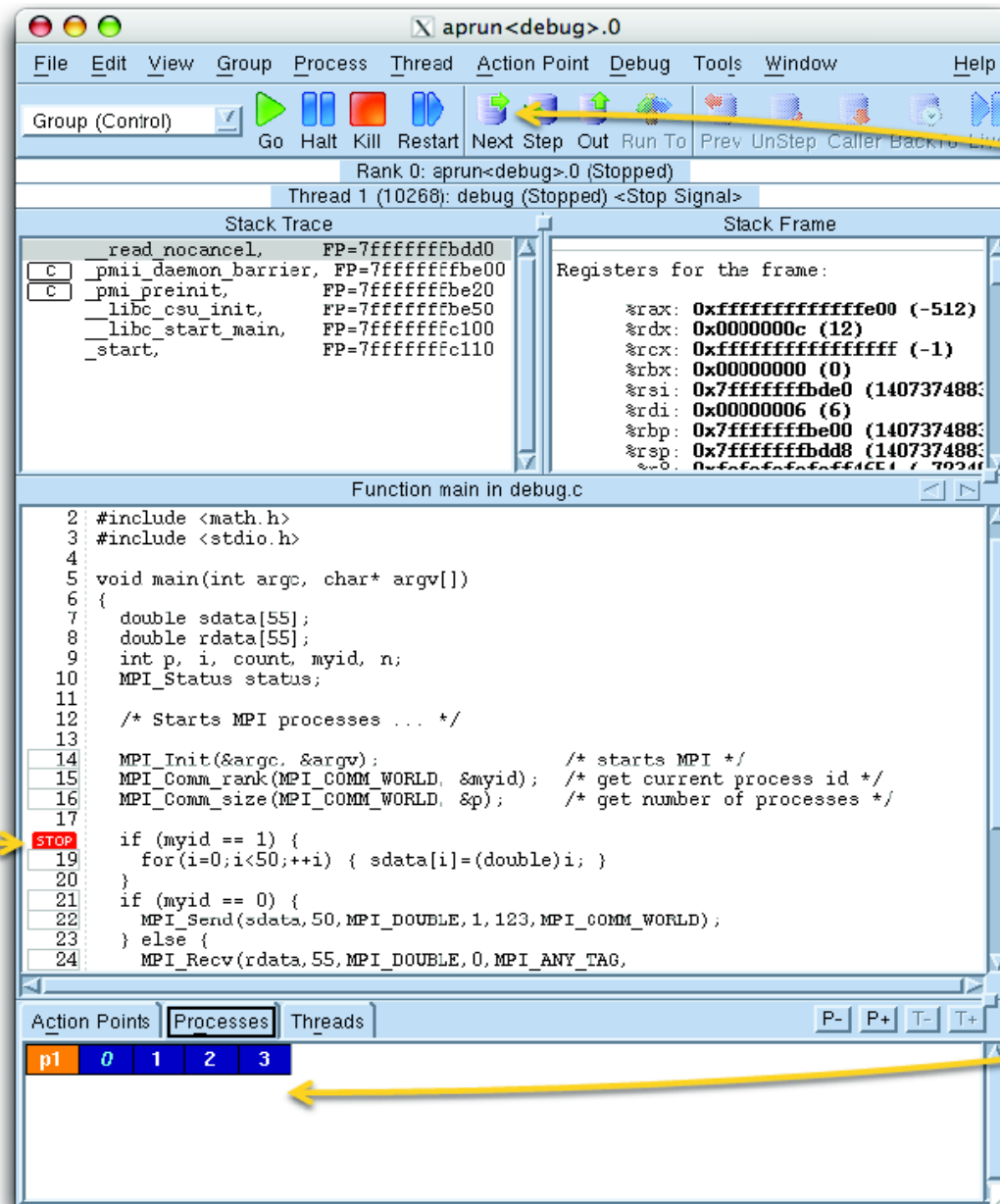


Debugging

Demos	OpenMP	MPI	OMP/MPI	Reference
gdb	yes	yes	no	http://www.gnu.org/s/gdb/
compiler	yes	yes	yes	-g -traceback
nemiver	no	yes	no	http://projects.gnome.org/nemiver/
eclipse	yes	no	no	http://www.eclipse.org
DDT	yes	yes	yes	http://www.allinea.com/ddt
Totalview	yes	yes	yes	http://www.roguewave.com

- A debugger can help find bugs in :
 - Fortran, C, C++, ...
 - Serial, MPI, OpenMP, MPI/OpenMP.
- A debugger can be executed :
 - either as a graphical user interface,
 - Or from a command-line interface
 - <http://www.open-mpi.org/faq/?category=debugging>

Breakpoints



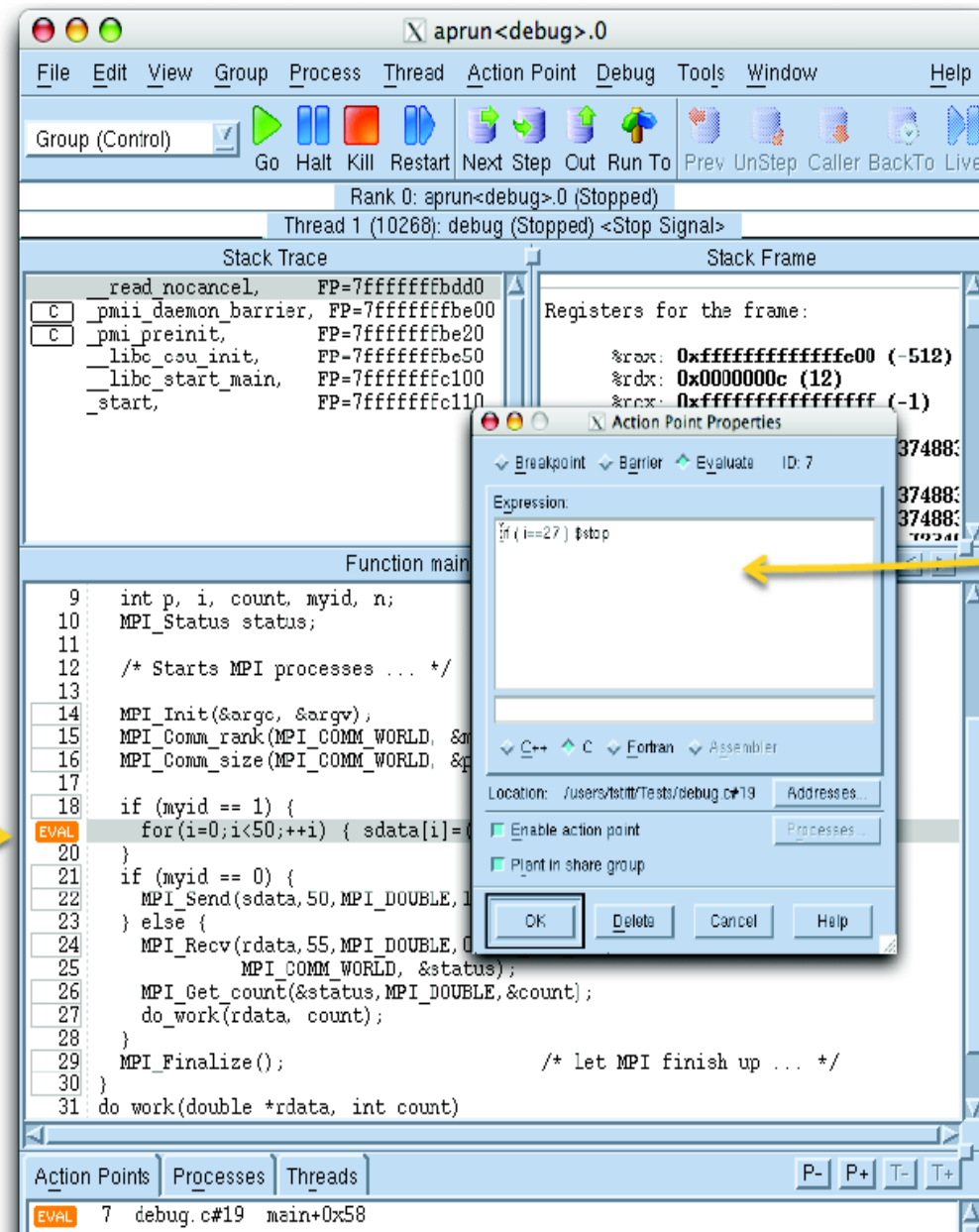
Set
Breakpoint

Execution
Toolbars

Co-operating
Processes

Action points

Set
Evaluation Point



Enter
Stopping
Condition

Examining data

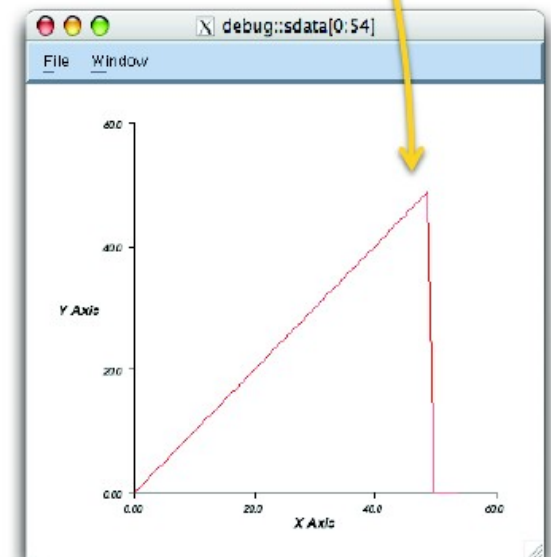
The screenshot shows the aprun debugger interface. The top menu includes File, Edit, View, Group, Process, Thread, Action Point, Debug, Tools, Window, and Help. Below the menu is a toolbar with buttons for Go, Halt, Kill, Restart, Next Step, Out, Run To, Prev, UnStep, Caller, and Back. The main window is divided into several panes:

- Rank 1:** aprun<debug>.1 (At Breakpoint 1)
- Thread 1 (9798):** debug (At Breakpoint 1)
- Stack Trace:** Shows the call stack with frames for main, _libc_start_main, and _start.
- Stack Frame:** Shows the current function "main" with local variables: sdata (double[55]), rdata (double[55]), p (0x00000004), i (0x00000032), count (0x00000032), myid (0x00000001), n (0x00000000), and status (MPI_Status).
- Function main in debug.c:** Shows the source code with line numbers 15 to 34. Line 19 is highlighted: `for(i=0; i<50; ++i) { sdata[i] = (double)i; }`
- Action Points:** Shows a breakpoint at debug.c#27, main+0x10a.

Dive in
And
Visualize

The screenshot shows the sdata variable viewer. The expression is sdata, the address is 0x70000000, and the type is double[55]. The data is displayed as a list of values from index 0 to 14:

Field	Value
[0]	0
[1]	1
[2]	2
[3]	3
[4]	4
[5]	5
[6]	6
[7]	7
[8]	8
[9]	9
[10]	10
[11]	11
[12]	12
[13]	13
[14]	14



Viewing data across processes

The screenshot shows a debugger window titled 'aprun <debug>.0' with a menu bar (File, Edit, View, Group, Process, Thread, Action Point, Debug, Tools, Window, Help) and a toolbar. The main window displays the source code of 'main.c' with a yellow highlight on the line 'if (myid == 1)'. A 'myid' variable is being inspected, showing its value across four processes (0, 1, 2, 3). A context menu is open over the 'myid' variable, with 'Dive into myid' selected. The 'Dive into myid' dialog shows a table of process IDs and values.

Process	Value
aprun<debug>.0	0x00000000 (0)
aprun<debug>.1	0x00000001 (1)
aprun<debug>.2	0x00000002 (2)
aprun<debug>.3	0x00000003 (3)

The 'Dive into myid' dialog also shows a list of actions: Dive, Dive in New Window, Dive in All, Expand All (Ctrl++), Collapse All (Ctrl+-), Undive, Undive All, Recive, Recive All, Freeze, Lock Address, Show Across, Compilation Scope, Loader Symbols, Padding, Break At Navlines, Examine Format, Lock Status. The 'Dive into myid' dialog also shows a list of actions: None (Ctrl+0), Processes (Ctrl+Shift+L), Threads (Ctrl+L).

Dive into *myid*

Memory usage

The screenshot shows a debugger interface with several windows. The main window displays a stack trace and source code for a function named 'main'. A breakpoint is set at line 18 of 'debug.c'. The 'Memory Debugging' window is open, showing a pie chart of memory usage for 'Process 2: aprun-debug-0'. The pie chart is divided into segments for Text (green), Data (red), Heap (blue), Stack (cyan), Stack VM (purple), and Total VM (yellow). A table below the pie chart provides the following data:

Process	Text	Data	Heap	Stack	Stack VM	Total VM
Process 2: aprun-debug-0	1207.16KB	506.22KB	12.04MB	12.50KB	92.00KB	23.90MB
Process 3: aprun-debug-1	1207.16KB	506.22KB	12.04MB	12.50KB	92.00KB	23.90MB
Process 4: aprun-debug-2	1207.16KB	506.22KB	12.04MB	12.50KB	92.00KB	23.90MB
Process 5: aprun-debug-3	1207.16KB	506.22KB	12.04MB	12.50KB	92.00KB	23.90MB

The 'Process 2: aprun-debug-0' legend shows the following breakdown:

- Text: 1207.16 KB (3.13%)
- Data: 506.22 KB (1.01%)
- Heap: 12.04 MB (31.92%)
- Stack: 12.50 KB (0.03%)
- Stack VM: 92.00 KB (0.24%)
- Total VM: 23.90 MB (63.67%)

The source code window shows the following code snippet:

```
6 {
7   double sdata[55];
8   double rdata[55];
9   int p, i, count, myid, n;
10  MPI_Status status;
11
12  /* Starts MPI processes ... */
13
14  MPI_Init(&argc, &argv);
15  MPI_Comm_rank(MPI_COMM_WORLD, &myid);
16  MPI_Comm_size(MPI_COMM_WORLD, &p);
17
18  if (myid == 1) {
19    for(i=0; i<50; ++i) { sdata[i] = (double) i; }
20  }
21  if (myid == 0) {
22    MPI_Send(sdata, 50, MPI_DOUBLE, 1, 12, MPI_COMM_WORLD);
23  } else {
24    MPI_Recv(rdata, 55, MPI_DOUBLE, 0, 12, MPI_COMM_WORLD, &status);
25    MPI_Get_count(&status, MPI_DOUBLE, &count);
26    MPI_Get_count(&status, MPI_DOUBLE, &count);
27    do_work(rdata, count);
28  }
```

View Memory Usage
At Breakpoint

Memory leaks

The screenshot shows a debugger interface with the following components:

- Process Set:** Lists processes including 'Parallel Job debug (MPI)' and 'aprun (31370)'.
- Stack Trace:** Shows the current call stack with 'main' at the top.
- Code Window:** Displays C code with a breakpoint set at line 19: `if (myid == 1) {`.
- Memory Debugging Window:** Contains several panes:
 - Options:** Includes 'Detect Leaks' and 'Relative to Baseline' checkboxes.
 - Graph:** A horizontal bar chart showing memory usage over time, with a yellow arrow pointing to a specific memory block.
 - Overall Totals Table:**

Category	Bytes	Count
Heap		
Allocated	2.09MB	55
Filtered	0	0
Unfiltered	2.09MB	55
Guard Blocks	1472	70
Post-guard	736	35
Pre-guard	736	35
Leaked	16	1
Unfiltered	16	1
Filtered	0	0
Corrupted Guard Blocks	0	0
Post-guard	0	0
Pre-guard	0	0
Deallocated	15.56KB	84
Filtered	0	0
Unfiltered	15.56KB	84
 - Selected Block Table:**

Property	Value
----------	-------
 - Related Blocks Table:**

Category	Bytes	Count
----------	-------	-------

View Memory Leaks At Breakpoint

Getting started (1)

```
2 program who
3 #ifdef _MPI
4   use mpi
5 #endif
6 !$   use omp_lib
7     implicit none
8     integer :: rank=-1,nb_procs=-1,code=-1,thread=-1,threads=-1
9     integer :: namelen=-1, coreid=-1
10    character(len=2) :: processor_name=""
11    integer, external :: running_on
12
13 #ifdef _MPI
14   call MPI_INIT (code)
15   call MPI_COMM_SIZE ( MPI_COMM_WORLD ,nb_procs,code)
16   call MPI_COMM_RANK ( MPI_COMM_WORLD ,rank,code)
17   call MPI_Get_processor_name( processor_name, namelen, code )
18 #endif
19
20 !$omp parallel private(thread)
21 !$   thread = omp_get_thread_num()
22 !$   threads = omp_get_num_threads()
23   coreid = running_on()
24   write (*, '(a17,i4,a2,i4,a5,i4,a2,i4,1x,a2,i4)') &
25     "hello in f90 rnk=", &
26     rank, " / ", nb_procs, &
27     " thd=", thread, " / ", threads, &
28     processor_name, &
29     coreid
30 !$omp end parallel
31
32 #ifdef _MPI
33   call MPI_FINALIZE (code)
34 #endif
35
36 end program who
```

```
mpif90 -D_MPI -fopenmp -g mpiomp.F90 -L/softs/affinity -laff -o f
```

```
n1:/home/piccinali/trunk/debug/intro/f90 $ sbatch.sh
USAGE :
      arg1=exe
      arg2=mppwidth
      arg3=mppnppn
      arg4=mppdepth
      arg5=exeargs
      arg6=prempiexec
      arg7=postmpiexec
```

```
n1:/home/piccinali/trunk/debug/intro/f90 $ export OMP_NUM_THREADS=3 ; ~/sbatch.sh ./f 2 2 3 "" "" -bind-to-core
+ export OMP_NUM_THREADS=3
+ OMP_NUM_THREADS=3
+ /usr/bin/time -p /softs/openmpi-1.4.3/bin/mpiexec -bind-to-core -n 2 -npnode 2 -x OMP_NUM_THREADS -hostfile /softs/openmpi-1.4.3/h ./f
```

```
n1:/home/piccinali/trunk/debug/intro/f90 $ cat o_f.0006.2.2.3.-bind-to-core
hello in f90 rnk= 1 / 2 thd= 0 / 3 n1 1
hello in f90 rnk= 1 / 2 thd= 2 / 3 n1 1
hello in f90 rnk= 1 / 2 thd= 1 / 3 n1 1
hello in f90 rnk= 0 / 2 thd= 0 / 3 n1 0
hello in f90 rnk= 0 / 2 thd= 2 / 3 n1 0
hello in f90 rnk= 0 / 2 thd= 1 / 3 n1 0
real 0.04
```


Getting started (2)

```
n1:/home/piccinali/trunk/debug/intro/f90 $ more o_f.0006.2.2.3.-bind-to-none o_f.0006.2.2.3.-bind-to-core o_f.0006.2.2.3.-bind-to-socket
:
o_f.0006.2.2.3.-bind-to-none
:
hello in f90 rnk= 1 / 2 thd= 0 / 3 n1 11
hello in f90 rnk= 0 / 2 thd= 0 / 3 n1 11
hello in f90 rnk= 1 / 2 thd= 2 / 3 n1 11
hello in f90 rnk= 1 / 2 thd= 1 / 3 n1 11
hello in f90 rnk= 0 / 2 thd= 2 / 3 n1 11
hello in f90 rnk= 0 / 2 thd= 1 / 3 n1 11
real 1.06
user 0.06
sys 0.04
:
o_f.0006.2.2.3.-bind-to-core
:
hello in f90 rnk= 1 / 2 thd= 0 / 3 n1 1
hello in f90 rnk= 1 / 2 thd= 2 / 3 n1 1
hello in f90 rnk= 1 / 2 thd= 1 / 3 n1 1
hello in f90 rnk= 0 / 2 thd= 0 / 3 n1 0
hello in f90 rnk= 0 / 2 thd= 2 / 3 n1 0
hello in f90 rnk= 0 / 2 thd= 1 / 3 n1 0
real 0.04
user 0.02
sys 0.02
:
o_f.0006.2.2.3.-bind-to-socket
:
hello in f90 rnk= 0 / 2 thd= 2 / 3 n1 5
hello in f90 rnk= 0 / 2 thd= 1 / 3 n1 5
hello in f90 rnk= 0 / 2 thd= 0 / 3 n1 5
hello in f90 rnk= 1 / 2 thd= 2 / 3 n1 5
hello in f90 rnk= 1 / 2 thd= 0 / 3 n1 5
hello in f90 rnk= 1 / 2 thd= 1 / 3 n1 5
real 0.05
user 0.03
sys 0.05

n1:/home/piccinali/trunk/debug/intro/f90 $ more o_f.0006.2.1.3.-bind-to-none o_f.0006.2.1.3.-bind-to-core o_f.0006.2.1.3.-bind-to-socket
:
o_f.0006.2.1.3.-bind-to-none
:
hello in f90 rnk= 0 / 2 thd= 0 / 3 n1 11
hello in f90 rnk= 0 / 2 thd= 2 / 3 n1 11
hello in f90 rnk= 0 / 2 thd= 1 / 3 n1 11
hello in f90 rnk= 1 / 2 thd= 0 / 3 n2 11
hello in f90 rnk= 1 / 2 thd= 2 / 3 n2 11
hello in f90 rnk= 1 / 2 thd= 1 / 3 n2 11
real 0.36
user 0.04
sys 0.03
:
o_f.0006.2.1.3.-bind-to-core
:
hello in f90 rnk= 0 / 2 thd= 0 / 3 n1 0
hello in f90 rnk= 0 / 2 thd= 2 / 3 n1 0
hello in f90 rnk= 0 / 2 thd= 1 / 3 n1 0
hello in f90 rnk= 1 / 2 thd= 0 / 3 n2 0
hello in f90 rnk= 1 / 2 thd= 2 / 3 n2 0
hello in f90 rnk= 1 / 2 thd= 1 / 3 n2 0
real 0.35
user 0.02
sys 0.04
:
o_f.0006.2.1.3.-bind-to-socket
:
hello in f90 rnk= 0 / 2 thd= 0 / 3 n1 5
hello in f90 rnk= 0 / 2 thd= 2 / 3 n1 5
hello in f90 rnk= 0 / 2 thd= 1 / 3 n1 5
hello in f90 rnk= 1 / 2 thd= 0 / 3 n2 5
hello in f90 rnk= 1 / 2 thd= 2 / 3 n2 5
hello in f90 rnk= 1 / 2 thd= 1 / 3 n2 5
real 0.39
user 0.05
```

Demo : GDB

Frequently used GDB commands

General Commands

- **help [name]** : Show information about GDB command
- **run [<args>]** : runs selected program with arguments <args>
- **attach <pid>** : attach gdb to a running process
- **Kill** : kills the process being debugged
- **Quit** : quits the gdb program

Stepping and Continuing

- **c[ontinue]** : continue execution (after a stop)
- **s[tep]** : step one line, entering called functions
- **n[ext]** : step one line, without entering functions

Breakpoint commands

- **b[reak] [<where>]** : sets breakpoints. <where> can be a function name, a line number or a hex address
- **[r]watch <expr>** : sets a watchpoint, which will break
 - when <expr> is written to [or read]
- **info break[points]** : prints out a listing of all breakpoints
- **d[elete] [<nums>]** : deletes breakpoints

Commands for looking around

- **list [<where>]** : prints out source code at <where>
- **backtrace [<n>]** : prints a backtrace <n> levels deep
- **info [<what>]** : prints out info on <what>
- **p[rint] [<expr>]** : prints out <expr>
- **d[isplay]** : prints value of expression each time the program stops.



Demo : gdb (mpi only)

```
nl:/home/piccinali/trunk/debug/intro/f90 $ mpif90 -D_MPI -g mpiomp.F90 -L/softs/affinity -laff -o f
```

```
/softs/openmpi-1.4.3/bin/mpixec -bind-to-core -n 2 -npnode 1 -x DISPLAY -hostfile /softs/openmpi-1.4.3/h xterm -rv -e gdb ./f
```

```
gdb
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/piccinali/trunk/debug/intro/f90/f...done.
(gdb) break 28
Breakpoint 1 at 0x400e0a: file mpiomp.F90, line 28.
(gdb) run
Starting program: /home/piccinali/trunk/debug/intro/f90/f
[Thread debugging using libthread_db enabled]
[New Thread 0x7ffff0f23700 (LWP 10166)]
[New Thread 0x7ffff0315700 (LWP 10167)]

Breakpoint 1, who () at mpiomp.F90:28
28                                coreid
(gdb) print rank
$1 = 0
(gdb) print nb_procs
$2 = 2
(gdb) █
```

Insert breakpoint
Start execution

View data
across processes

```
gdb
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/piccinali/trunk/debug/intro/f90/f...done.
(gdb) break 28
Breakpoint 1 at 0x400e0a: file mpiomp.F90, line 28.
(gdb) run
Starting program: /home/piccinali/trunk/debug/intro/f90/f
[Thread debugging using libthread_db enabled]
[New Thread 0x7ffff0f32700 (LWP 32165)]
[New Thread 0x7ffff0324700 (LWP 32166)]

Breakpoint 1, who () at mpiomp.F90:28
28                                coreid
(gdb) print rank
$1 = 1
(gdb) print nb_procs
$2 = 2
(gdb) █
```

```
2 program who
3 #ifdef _MPI
4     use mpi
5 #endif
6 !$   use omp_lib
7     implicit none
8     integer :: rank=-1,nb_procs=-1,code=-1,thread=-1,threads=-1
9     integer :: namelen=-1, coreid=-1
10    character(len=2) :: processor_name=""
11    integer, external :: running_on
12
13 #ifdef _MPI
14    call MPI_INIT (code)
15    call MPI_COMM_SIZE ( MPI_COMM_WORLD ,nb_procs,code)
16    call MPI_COMM_RANK ( MPI_COMM_WORLD ,rank,code)
17    call MPI_Get_processor_name( processor_name, namelen, code )
18 #endif
19
20 !$omp parallel private(thread)
21 !$   thread = omp_get_thread_num()
22 !$   threads = omp_get_num_threads()
23     coreid = running_on()
24     write (*,'(a17,i4,a2,i4,a5,i4,a2,i4,1x,a2,i4)') &
25         "hello in f90 rnk=",&
26         rank," /",nb_procs,&
27         " thd=",thread," /",threads, &
28         processor_name, &
29         coreid
30 !$omp end parallel
```

Demo : gdb (openmp only)

```
nl:/home/piccinali/trunk/debug/intro/f90 $ gfortran -fopenmp -g mpiomp.F90 -L/softs/affinity -laff -o f
nl:/home/piccinali/trunk/debug/intro/f90 $ export OMP_NUM_THREADS=2 ; gdb ./f
```

```
GNU gdb (Ubuntu/Linaro 7.2-1ubuntu11) 7.2
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/piccinali/trunk/debug/intro/f90/f...done.
```

Insert breakpoint
Start execution

```
(gdb) break 26
Breakpoint 1 at 0x400ac4: file mpiomp.F90, line 26.
(gdb) run
Starting program: /home/piccinali/trunk/debug/intro/f90/f
[Thread debugging using libthread_db enabled]
[New Thread 0x7ffff6c94700 (LWP 10501)]
```

```
Breakpoint 1, MAIN__omp_fn.0 (.omp_data_i=0x0) at mpiomp.F90:26
26      " thd=",thread," /",threads, &
```

```
(gdb) print thread
$1 = 0
```

```
(gdb) info thread
```

```
2 Thread 0x7ffff6c94700 (LWP 10501) __lll_lock_wait () at ../nptl/sysdeps/unix/sysv
* 1 Thread 0x7ffff7fe5780 (LWP 10498) MAIN__omp_fn.0 (.omp_data_i=0x0) at mpiomp.F90
```

```
(gdb) thread 2
```

```
[Switching to thread 2 (Thread 0x7ffff6c94700 (LWP 10501))]#0 __lll_lock_wait () at .
136      ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S: No such file or directo
in ../nptl/sysdeps/unix/sysv/linux/x86_64/lowlevellock.S
```

```
(gdb) cont
```

```
Continuing.
```

```
hello in f90 rnk= -1 / -1 thd= 0 / 2 11
```

```
[Switching to Thread 0x7ffff6c94700 (LWP 10501)]
```

```
Breakpoint 1, MAIN__omp_fn.0 (.omp_data_i=0x0) at mpiomp.F90:26
26      " thd=",thread," /",threads, &
```

```
(gdb) print thread
```

```
$2 = 1
```

```
(gdb) cont
```

```
Continuing.
```

```
hello in f90 rnk= -1 / -1 thd= 1 / 2 11
```

```
[Thread 0x7ffff6c94700 (LWP 10501) exited]
```

```
Program exited normally.
```

```
(gdb) q
```

View data
across threads

```
2 program who
3 #ifdef _MPI
4     use mpi
5 #endif
6 !$
7     use omp_lib
8     implicit none
9     integer :: rank=-1,nb_procs
10    integer :: namelen=-1, co
11    character(len=2) :: proces
12    integer, external :: runni
13 #ifdef _MPI
14    call MPI_INIT (code)
15    call MPI_COMM_SIZE ( MPI_CO
16    call MPI_COMM_RANK ( MPI_CO
17    call MPI_Get_processor_name
18 #endif
19
20 !$omp parallel private(thread)
21 !$     thread = omp_get_thread_num
22 !$     threads = omp_get_num_thre
23     coreid = running_on()
24     write (*,'(a17,i4,a2,i4,a5,
25            "hello in f90 rnk=",
26            rank," /",nb_procs,
27            " thd=",thread," /",
28            processor_name, &
29            coreid
30 !$omp end parallel
31
32
33 #ifdef _MPI
34     call MPI_FINALIZE (code)
35 #endif
36
37 end program who
```

Demo : Nemiver

ANGD 10/2011

Demo : nemiver (mpi only)

```
n1:/home/piccinali/trunk/debug/intro/f90 $ mpif90 -D_MPI -g mpiomp.F90 -L/softs/affinity -laff -o f
```

```
n1:/home/piccinali/trunk/debug/intro/f90 $ xhost
access control enabled, only authorized clients can connect
INET:n4.local
INET:n3.local
INET:n1.local
INET:n2.local
```

```
n1:/home/piccinali/trunk/debug/intro/f90 $ echo $DISPLAY
n1:1
```

```
/softs/openmpi-1.4.3/bin/mpirun -n 2 -npernode 1 -x DISPLAY -hostfile /softs/openmpi-1.4.3/h nemiver ./f
```

Start execution

Insert breakpoint

Thread ID	Frame	Function	Arguments	Variable	Value
3	0	who	()		
2	1	main	(argc = 1, argv = 0x7fffffe14e !/home/piccinali/trunk/debug/intro/f90/f?)	code	0
1	2	__libc_start_main	(main = 0x400f6c <main>, argc = 1, ubp_av = 0x7fffffd2e18, init = <value>	coreid	11
	3	_start	()	namelen	2
				nb_procs	2
				processor_name	'n1'
				rank	0
				thread	-1
				threads	-1

View data across processes

Thread ID	Frame	Function	Arguments	Location	Address	Binary	Variable	Value
3	0	who	()	mpiomp.F90:28	0x0000000000400e0a			
2	1	main	()	mpiomp.F90:3	0x0000000000400fa5		code	0
1	2	__libc_start_main	()		0x00007ffff5f20eff	/lib/x86_64-linux-	coreid	11
	3	_start	()		0x0000000000400ce9		namelen	2
							nb_procs	2
							processor_name	'n2'
							rank	1
							thread	-1
							threads	-1

Demo : Eclipse

ANGD 10/2011

Demo : eclipse (openmp only)

Start execution

Insert breakpoint

```
8  
9 program helloompf  
10 !$ use omp lib  
11 implicit none  
12 integer:: thread=-1, threads=-1  
13  
14 !$omp parallel private(thread)  
15 thread = omp_get_thread_num()  
16 threads = omp_get_num_threads()  
17 write (*,'(i4,a13,i4)') thread," hello in f90", threads  
18 !$omp end parallel  
19  
20 ! print *, "Hello MPI World"  
21 end program  
22
```

View data across processes

Name	Value
.omp_data_i <mi_cmd_var_create: ur	
thread	0

Name	Value
.omp_data_i <mi_cmd_var_create: ur	
thread	1

Demo : DDT

(thanks for the demo license)

Demo : ddt (impi/omp)

```
/softs/openmpi-1.4.3/bin/mpirun -debug -n 2 -npnode 1 -x DISPLAY -x OMP_NUM_THREADS -hostfile /softs/openmpi-1.4.3/h ./f
```

Start execution

Insert breakpoint

View data across processes

Session Control Search View Help

Current Group: All Focus on current: Group Process Thread Step Threads Together

Process 1's threads: 1 2 3

Proj... Fortran ... f mpiomp.F90

```
1 program who
2 #ifdef MPI
3 use mpi
4 #endif
5 !$ use omp lib
6 implicit none
7 integer :: rank=-1,nb_procs=-1,code=-1,thread=-1,threads=-1
8 integer :: namelen=-1, coreid=-1
9 character(len=2) :: processor_name=""
10 integer, external :: running_on
11
12 #ifdef _MPI
13 call MPI_INIT (code)
14 call MPI_COMM_SIZE ( MPI_COMM_WORLD , nb_procs,code)
15 call MPI_COMM_RANK ( MPI_COMM_WORLD , rank,code)
16 call MPI_Get_processor_name( processor_name, namelen, code )
17 #endif
18
19 !$omp parallel private(thread)
20 !$ thread = omp_get_thread_num()
21 !$ threads = omp_get_num_threads()
22 coreid = running_on()
23 write (*, '(a17,i4,a2,i4,a5,i4,a2,i4,1x,a2,i4)') &
24 "hello in f90 rnk=", &
25 rank, " / ", nb_procs, &
26 " thd=", thread, " / ", threads, &
27 processor_name, &
28 coreid
29 !$omp end parallel
30
31
32 #ifdef MPI
33 call MPI_FINALIZE (code)
34 #endif
35
36 end program who
```

Current Stack

Stack Arguments

```
#0 who () at /home/piccinal/trunk/debug/intro/f90/mpiomp.F90:16
```

Current Line(s)

Variable Name	Value
processor_name	' '
namelen	-1
code	0

Type: none selected

Locals

Variable Name	Value
code	0
coreid	-1
namelen	-1
nb_procs	2
processor_name	' '
rank	1
thread	-1
threads	-1

Type: none selected

Input/Output Breakpoints (Process 1) Watchpoints (Process 1) Tracepoints (Process 1) Stacks (Process 1)

Input/Output

Output For Rank: All

```
mpirun: hello in f90 rnk= 0 / 2 thd= 2 / 3 n1 11
mpirun: hello in f90 rnk= 0 / 2 thd= 0 / 3 n1 11
mpirun: hello in f90 rnk= 0 / 2 thd= 1 / 3 n1 11
```

Note: DDT can only send input to the mpirun process with this MPI

Type here ('Enter' to send):

ANGD 10/2011

```
sudo sysctl -w kernel.yama.ptrace_scope=0
```

Demo : Totalview

ANGD 10/2011

Launching Totalview (1)

```

palu1: salloc -N2
salloc: Granted job allocation 2960
palu1: export OMP_NUM_THREADS=2 ; totalview aprun -a -n2 -N1 -d24 exe
Linux x86_64 TotalView 8.8.0-1
Copyright 2007-2010 by TotalView Technologies, LLC. ALL RIGHTS RESERVED.
Copyright 1999-2007 by Etnus, LLC.
Copyright 1999 by Etnus, Inc.
Copyright 1996-1998 by Dolphin Interconnect Solutions, Inc.
Copyright 1989-1996 by BBN Inc.
TotalView Technologies ReplayEngine
Copyright 2010 TotalView Technologies
ReplayEngine uses the UndoDB Reverse Execution Engine
Copyright 2005-2010 Undo Limited
Reading symbols for process 1, executing "aprun"
Library /usr/bin/aprun, with 2 asects, was linked at 0x00400
t 0xff00000090000000
Mapping 25131 bytes of ELF string data from '/usr/bin/aprun
Indexing 26800 bytes of DWARF '.debug_frame' symbols from '
Indexing 28044 bytes of DWARF '.eh_frame' symbols from '/usr
Skimming 181212 bytes of DWARF '.debug_info' symbols from '
Library @syscall_library@-64, with 1 asects, was linked at (
ially loaded at 0xff00000090009d800
INFO: Using previously cached local copy of library "@sysca
Mapping 82 bytes of ELF string data from '@syscall_library@
Indexing 192 bytes of DWARF '.eh_frame' symbols from '@sysca
Library /lib64/libdl.so.2, with 2 asects, was linked at 0x00
d at 0xff00000090009e100
Mapping 459 bytes of ELF string data from '/lib64/libdl.so.2
Indexing 620 bytes of DWARF '.eh_frame' symbols from '/lib64
Library /lib64/libpthread.so.0, with 2 asects, was linked at
loaded at 0xff0000009000a0500
Mapping 14508 bytes of ELF string data from '/lib64/libpthre
Indexing 12804 bytes of DWARF '.eh_frame' symbols from '/lib
Library /lib64/libc.so.6, with 2 asects, was linked at 0x000
at 0xff0000009000baa00
Mapping 21676 bytes of ELF string data from '/lib64/libc.so
Indexing 126700 bytes of DWARF '.eh_frame' symbols from '/l
Library /lib64/ld-linux-x86-64.so.2, with 2 asects, was link
ally loaded at 0xff00000090007ee00
Mapping 380 bytes of ELF string data from '/lib64/ld-linux-x
Indexing 6072 bytes of DWARF '.eh_frame' symbols from '/lib6
one

```



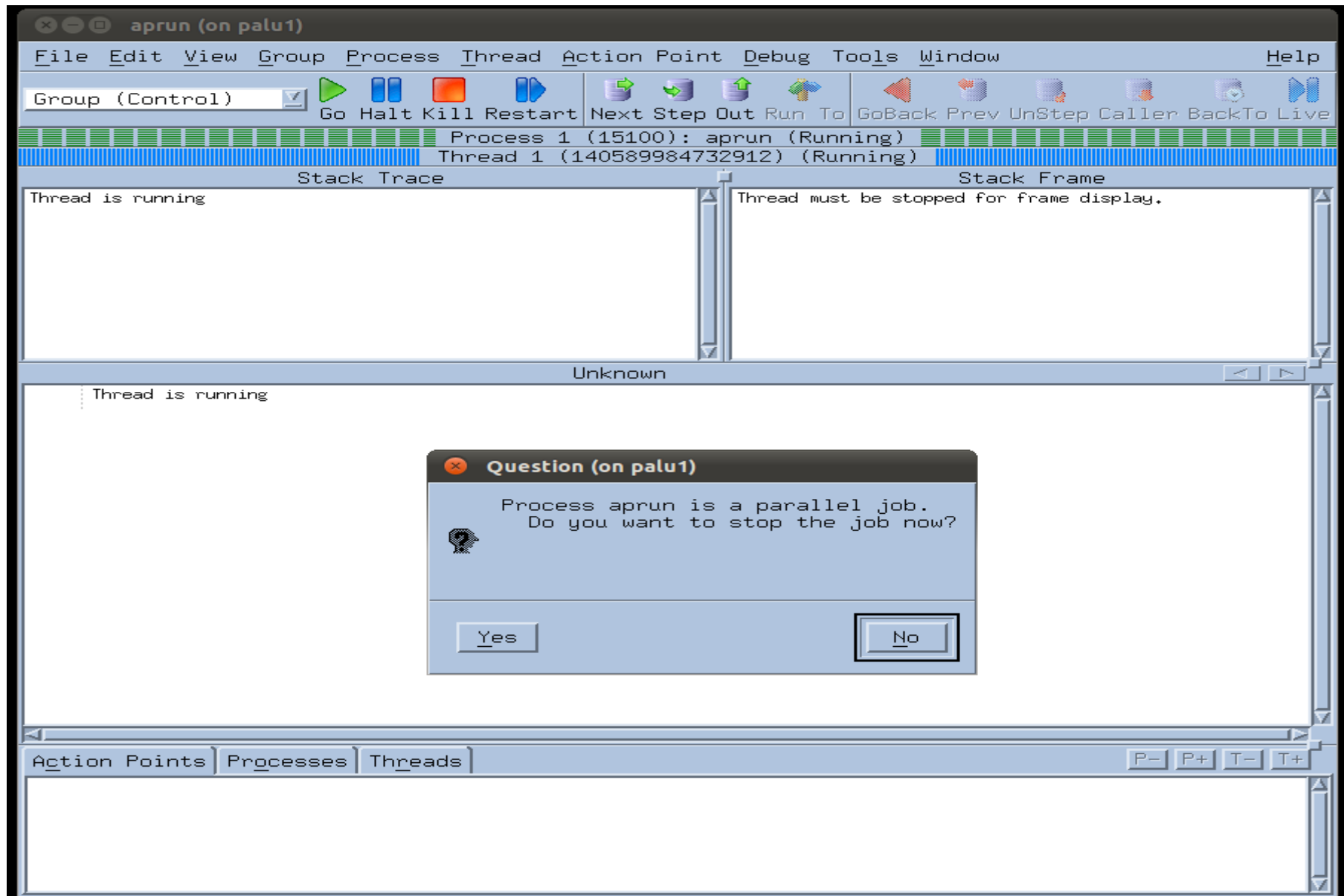
File Edit View Tools Window Help				
ID	Rank	Host	Status	Description
... 1		<local>	-	aprun (0 active threads)

The screenshot shows the TotalView GUI with the following components:

- Root window:** Displays process and thread information.
- Process window:** Displays stack trace, stack frame, and source code for the selected thread in a process.
- Process group window:** Displays process groups for multiprocess programs.
- Variable window:** Displays address, data type, and value of local variable, register, or global variable.

ANGD 10/2011

Launching Totalview (2)



Inserting Breakpoints

The screenshot displays a debugger window with the following components:

- Menu Bar:** File, Edit, View, Group, Process, Thread, Action Point, Debug, Tools, Window, Help.
- Toolbar:** Go, Halt, Kill, Restart, Next Step, Out, Run, To, GoBack, Prev, UnStep, Caller, BackTo, Live.
- Status Bar:** Rank 0: aprun<exe>.0 (At Breakpoint 1); Thread 1 (5614): exe (At Breakpoint 1).
- Stack Trace:**
 - f90 MAIN__omp_fn.0
 - f90 affinity
 - f90 main
 - c __libc_start_main
 - c _start
- Stack Frame:**
 - Function "MAIN__omp_fn.0":
 - Local variables:
 - nid: ""
 - coreid: 0 (0x00000000)
 - threadid: 0 (0x00000000)
 - Registers for the frame:
 - %rax: 0x00000002 (2)
 - %rdx: 0x00894b20 (8997664)
 - %rcx: 0x004c027e (4981374)
- Source Code:** Function MAIN__omp_fn.0 in ...

```
18 integer :: coreid
19 integer :: rc=0, rank=0, numtask=0, coresperrcn=24
20 integer:: threadid=0, nthreads=0
21
22 #ifdef _MPI
23 call MPI_INIT ( rc )
24 call MPI_COMM_RANK ( MPI_COMM_WORLD, rank, rc )
25 call MPI_COMM_SIZE ( MPI_COMM_WORLD, numtask, rc )
26 #endif
27
28 !omp parallel private(threadid,coreid,nid)
29 !$ threadid = omp_get_thread_num()
30 !$ nthreads = omp_get_num_threads()
31 !! coresperrcn = omp_get_num_procs()
32 !! MAXT = OMP_GET_MAX_THREADS()
33 !! INPAR = OMP_IN_PARALLEL()
34 !! DYNAMIC = OMP_GET_DYNAMIC()
35 !! NESTED = OMP_GET_NESTED()
36
37 call print_nodeaffinity(nid)
38 ! call print_coreaffinity(coreid)
39 coreid = running_on()
40 ! barrier
```
- Process/Thread Table:**

ID	Rank	Host	Status	Description
1		<local>	R	aprun (1 active threads)
2	0	nid00003	B	aprun<exe>.0 (3 active threads)
2.1	0	nid00003	B1	in MAIN__omp_fn.0
2.2	0	nid00003	T	in __ioctl
2.3	0	nid00003	T	in MAIN__omp_fn.0
3	1	nid00004	B	aprun<exe>.1 (3 active threads)
3.1	1	nid00004	B1	in MAIN__omp_fn.0
3.2	1	nid00004	T	in __ioctl
3.3	1	nid00004	T	in MAIN__omp_fn.0
- Action Points:**

ID	Process	Thread
2.1	(5614) B1	in MAIN__omp_fn.0
2.2	(5659) T	in __ioctl
2.3	(5660) T	in MAIN__omp_fn.0

Viewing data across processes and threads

The screenshot shows the aprun debugger interface. The main window displays the stack trace for the current thread (Thread 1 (5732): exe (At Breakpoint 2)). The stack trace shows the following frames:

- f90 affinity.
- f90 main.
- C __libc_start_main.
- C _start.

The stack frame for the current function, "affinity", is expanded to show local variables:

- coreid: 0 (0x00000000)
- nid: ""
- nthreads: 2 (0x00000002)
- numtask: 2 (0x00000002)
- rank: 0 (0x00000000)
- rc: 0 (0x00000000)
- threadid: 0 (0x00000000)
- Modules:

The source code for the "affinity" function is visible in the main window, showing the following code:

```
30 |# nthreads = omp_get_num_threads()
31 || corespercn = omp_get_num_procs()
32 || MAXT = OMP_GET_MAX_THREADS()
33 || INPAR = OMP_IN_PARALLEL()
34 || DYNAMIC = OMP_GET_DYNAMIC()
35 || NESTED = OMP_GET_NESTED()
36
37 | call print_nodeaffinity(nid)
38 | call print_coreaffinity(coreid)
39 | coreid = running_on()
40 | barrier
41 | print *, &
42 | "rank=",rank,"/" numtask
43 | " cnid=",nid
44 | " threadid=",threadid
45 | " core=",coreid
46 |!omp end parallel
47 |#ifdef _MPI
48 | call MPI_FINALIZE()
49 |#endif
50 | print *, "done"
51 |
52 | end program
```

The expression window shows the evaluation of the variable "rank". The expression is "rank", the address is "0x007b1e94", and the type is "integer(kind=4)". The results table shows the value of "rank" for each process:

Process	Value
aprun<exe>.0	0 (0x00000000)
aprun<exe>.1	1 (0x00000001)

The expression window shows the evaluation of the variable "threadid". The expression is "threadid", the address is "0x007b1e9c", and the type is "integer(kind=4)". The results table shows the value of "threadid" for each thread:

Thread	Value
6.1 (5732)	0 (0x00000000)
6.2 (5778)	0 (0x00000000)

Restarting and exiting Totalview

```
Application 108251 resources: utime ~0s, stime ~0s
The TotalView Debugger Server has died
palu1: export OMP_NUM_THREADS=2 ; totalview aprun -a -n2 -N1 -d24 exe
Linux x86_64 TotalView 8.8.0-1
Copyright 2007-2010 by TotalView Technologies, LLC. ALL RIGHTS RESERVED.
Copyright 1999-2007 by Etnus, LLC.
Copyright 1999 by Etnus, Inc.
Copyright 1996-1998 by Dolphin Interconnect Solutions, Inc.
Copyright 1989-1996 by BBN Inc.
TotalView Technologies ReplayEngine
Copyright 2010 TotalView Technologies
ReplayEngine uses the UndoDB Reverse Execution Engine
Copyright 2005-2010 Undo Limited
Reading symbols for process 1, executing "aprun"
Library /usr/bin/aprun, with 2 asects, was linked at 0x00400000, and initially loaded at 0xff000000900000
Mapping 25131 bytes of ELF string data from '/usr/bin/aprun'...done
Indexing 26800 bytes of DWARF '.debug_frame' symbols from '/usr/bin/aprun'...done
Indexing 28044 bytes of DWARF '.eh_frame' symbols from '/usr/bin/aprun'...done
Skimming 181212 bytes of DWARF '.debug_info' symbols from '/usr/bin/aprun'...done
Library @syscall_library@-64, with 1 asects, was linked at 0xffffffff7000000, and initially loaded at 0x
INFO: Using previously cached local copy of library "@syscall_library@-64"
Mapping 82 bytes of ELF string data from '@syscall_library@-64'...done
Indexing 192 bytes of DWARF '.eh_frame' symbols from '@syscall_library@-64'...done
Library /lib64/libdl.so.2, with 2 asects, was linked at 0x00000000, and initially loaded at 0xff000000900
Mapping 459 bytes of ELF string data from '/lib64/libdl.so.2'...done
Indexing 620 bytes of DWARF '.eh_frame' symbols from '/lib64/libdl.so.2'...done
Library /lib64/libpthread.so.0, with 2 asects, was linked at 0x00000000, and initially loaded at 0xff00000
Mapping 14508 bytes of ELF string data from '/lib64/libpthread.so.0'...done
Indexing 12804 bytes of DWARF '.eh_frame' symbols from '/lib64/libpthread.so.0'...done
Library /lib64/libc.so.6, with 2 asects, was linked at 0x00000000, and initially loaded at 0xff000000900b
Mapping 21676 bytes of ELF string data from '/lib64/libc.so.6'...done
Indexing 126700 bytes of DWARF '.eh_frame' symbols from '/lib64/libc.so.6'...done
Library /lib64/ld-linux-x86-64.so.2, with 2 asects, was linked at 0x00000000, and initially loaded at 0xf
Mapping 380 bytes of ELF string data from '/lib64/ld-linux-x86-64.so.2'...done
Indexing 6072 bytes of DWARF '.eh_frame' symbols from '/lib64/ld-linux-x86-64.so.2'...done
Library /lib64/libnss_files.so.2, with 2 asects, was linked at 0x00000000, and initially loaded at 0xff00
Mapping 2007 bytes of ELF string data from '/lib64/libnss_files.so.2'...done
Indexing 3260 bytes of DWARF '.eh_frame' symbols from '/lib64/libnss_files.so.2'...done
Launching TotalView Debugger Servers with command:
svrlaunch /opt/toolworks/totalview.8.8.0a/linux-x86-64/bin/tvdsvrmain '-verbosity info' 172.26.0.31
INFO: Using previously cached local copy of library "/dsl/var/spool/alps/108253/exe"
Library /dsl/var/spool/alps/108253/exe, with 2 asects, was linked at 0x00400000, and initially loaded at
Mapping 132443 bytes of ELF string data from '/dsl/var/spool/alps/108253/exe'...done
Indexing 79272 bytes of DWARF '.debug_frame' symbols from '/dsl/var/spool/alps/108253/exe'...done
Indexing 115132 bytes of DWARF '.eh_frame' symbols from '/dsl/var/spool/alps/108253/exe'...done
Skimming 1646889 bytes of DWARF '.debug_info' symbols from '/dsl/var/spool/alps/108253/exe'...done
Reading symbols for process 2, executing "./exe"
Reading symbols for process 3, executing "./exe"
rank= 0 / 2 cnid=nid00003 threadid= 0 / 2 core= 0
rank= 0 / 2 cnid=nid00003 threadid= 1 / 2 core= 1
rank= 1 / 2 cnid=nid00004 threadid= 0 / 2 core= 0
rank= 1 / 2 cnid=nid00004 threadid= 1 / 2 core= 1
done
done
Application 108253 resources: utime ~0s, stime ~0s
palu1: exit
exit
salloc: Relinquishing job allocation 2960
salloc: Job allocation 2960 has been revoked.
palu1: █
```

ANGD 10/2011

Example 1 : Laplace

```
n1:/home/piccinali/trunk/debug/laplace $ make
mpif90 -O3 -g -c Laplace_mpi.F90
mpif90 -O3 -g -o f Laplace_mpi.o
GNU fortran executable ready
```

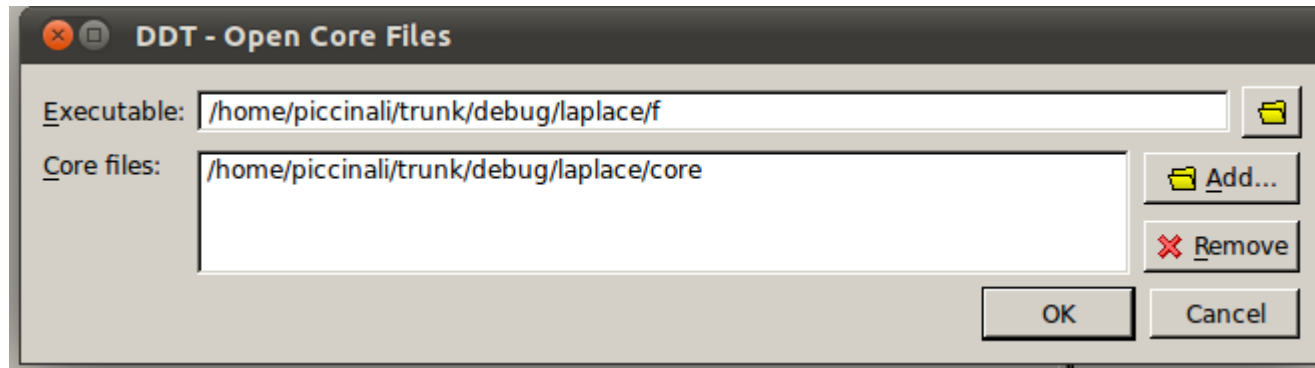
```
n1:/home/piccinali/trunk/debug/laplace $ /home/piccinali/sbatch.sh ./f 12 12 1 "1920 1920 10 1.0d-5" "" -bind-to-core
core file size      (blocks, -c) unlimited
data seg size      (kbytes, -d) unlimited
scheduling priority (-e) 20
file size          (blocks, -f) unlimited
pending signals    (-i) 16382
max locked memory  (kbytes, -l) unlimited
max memory size    (kbytes, -m) unlimited
open files         (-n) 1024
pipe size          (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size         (kbytes, -s) unlimited
cpu time           (seconds, -t) unlimited
max user processes (-u) unlimited
virtual memory     (kbytes, -v) unlimited
file locks         (-x) unlimited
+ export OMP_NUM_THREADS=1
+ OMP_NUM_THREADS=1
+ /usr/bin/time -p /softs/openmpi-1.4.3/bin/mpixec -bind-to-core -n 12 -npnode 12 -x OMP_NUM_THREADS -hostfile /softs/openmpi-1.4.3/h ./f 1920 1920 10 1.0d-5
```

```
n1:/home/piccinali/trunk/debug/laplace $ tail o_f.0012.12.12.1.1920-1920-200-1.0d-5--bind-to-core
[n1:21918] [ 4] /lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xff) [0x2b1310cffe]
[n1:21918] [ 5] ./f() [0x401429]
[n1:21918] *** End of error message ***
-----
mpixec noticed that process rank 6 with PID 21917 on node n1 exited on signal 11 (Segmentation fault).
-----
Command exited with non-zero status 139
```

Exemple 1 : Laplace (core file)

```
n1:/home/piccinali/trunk/debug/laplace $ file core
core: ELF 64-bit LSB core file x86-64, version 1 (SYSV), SVR4-style, from './f 1920 1920 10 1.0d-5'
```

```
n1:/home/piccinali/trunk/debug/laplace $ gdb f core
Core was generated by `./f 1920 1920 10 1.0d-5'.
Program terminated with signal 11, Segmentation fault.
#0  init_solver (dxx=Cannot access memory at address 0xfe7fb0
) at Laplace_mpi.F90:377
377          ve(i,j) = 2.0d0 * eps(i,j) * eps(i+1,j) / ( (eps(i,j) + eps(i+1,j)) * dxx**2)
```



Recompiler avec -fbacktrace

Exemple 1 : Laplace (DDT debugger)

```
/softs/openmpi-1.4.3/bin/mpixec -debug -bind-to-core -n 12 -npnode 12 -hostfile /softs/openmpi-1.4.3/h ./f 1920 1920 10 1.0d-5
```

Distributed Debugging Tool

Current Group: All | Focus on current: Group | Process | Thread | Step Threads Together

All | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11

Create Group

Proj... | Fortran ... | f Laplace_mpi.F90 x

Project Files | Search

Project Files

- Source Tree
- Header Files
- Source Files
- Laplace_mpi.F90
- ompi_debuggers

```
356 Integer, INTENT(IN) :: imin,imax,jmin,jmax
357 Real(rp), INTENT(IN) :: dxx, dyy
358 Real(rp), INTENT(OUT) :: omega
359 Real(rp), INTENT(OUT) :: omegal
360 Real(rp), INTENT(IN), Dimension(:,:) :: eps
361 Real(rp), INTENT(INOUT), Dimension(:,:) :: ve, vn, vs, vw, vc, sou
362 Integer :: i,j
363 !-----
364 ! Resolution de l'equation de Laplace
365 !-----
366
367 ! choix de la methode de resolution
368 ! omega = 1 : Gauss-Seidel
369 ! 1<omega<2 : Sur-relaxation
370 omega = 1.5d0
371 omegal = 1.0d0 - omega
372
373 ! Definition des coefficients de l'equation de Laplace
374 ! cas general
375 do j = jmin, jmax
376 do i = imin, imax
377 ve(i,j) = 2.0d0 * eps(i,j) * eps(i+1,j) / ( eps(i,j) + eps(i+1,j) ) * dxx**2
378 vn(i,j) = 2.0d0 * eps(i,j) * eps(i,j+1) / ( eps(i,j) + eps(i,j+1) ) * dyy**2
379 vs(i,j) = 2.0d0 * eps(i,j) * eps(i,j-1) / ( eps(i,j) + eps(i,j-1) ) * dyy**2
380 vw(i,j) = 2.0d0 * eps(i,j) * eps(i-1,j) / ( eps(i,j) + eps(i-1,j) ) * dxx**2
381 vc(i,j) = omega / ( vw(i,j) + vn(i,j) + vs(i,j) + ve(i,j) )
382 sou(i,j) = 0.0d0
383 enddo
384 enddo
385
386 end subroutine init_solver
387
388 subroutine solver(omegal, V,V_old, sou,ve,vw,vn,vs,vc, imin,imax,jmin,jmax, j)
389
390 implicit none
391 Integer, Parameter :: rp = Kind (1.0d0)
392 Real(rp), INTENT(INOUT), Dimension(imin-1:imax+1,jmin-1:jmax+1) :: V
393 Real(rp), INTENT(IN), Dimension(imin-1:imax+1,jmin-1:jmax+1) :: V_old
394 Real(rp), INTENT(IN) :: omegal
395 Real(rp), INTENT(IN), Dimension(imin:imax,jmin:jmax) :: sou, ve, vw, vn, vs, vc
396 Integer, INTENT(IN) :: imin,imax,j,jmin,jmax
397 Integer :: i, istat
398
399 do i = imin, imax
```

Current Stack

Stack Arguments

```
#1 laplace () at /home/piccinali/trunk/debug/laplace/Laplace_mpi.F90:212 (at 0x00000000000402
#0 init_solver (dxx=0.00052083333333333333, dyy=0.00052083333333333333, eps=<value
```

Distributed Debugging Tool

Processes 0-11:

Process stopped in init_solver (Laplace_mpi.F90:377) with signal SIGSEGV (Segmentation fault). Reason/Origin: the kernel Your program will probably be terminated if you continue. You can use the stack controls to see what the process was doing at the time.

Always show this window for signals

Continue Pause

Variable Name	Value
dxx	0.00052083333333333333
dyy	0.00052083333333333333
eps	<value optimized out>
ve	<value optimized out>
vn	<value optimized out>
vs	<value optimized out>
vw	<not allocated>
vc	<not allocated>
sou	<not allocated>
imin	1
imax	480
jmin	1
jmax	640
omega	1.5
omegal	-0.5
eps.0	<unknown bounds>
i	1

Type: none selected

Exemple 1 : Laplace (solution)

Dans la subroutine `init_solver`, les dimensions des tableaux doivent être déclarées :

```
Real(rp), INTENT(IN), Dimension(imin-1:imax+1,jmin-1:jmax+1) :: eps
```

```
Real(rp), INTENT(INOUT), Dimension(imin:imax,jmin:jmax) :: ve
```

```
Real(rp), INTENT(INOUT), Dimension(imin:imax,jmin:jmax) :: vn
```

```
Real(rp), INTENT(INOUT), Dimension(imin:imax,jmin:jmax) :: vs
```

```
Real(rp), INTENT(INOUT), Dimension(imin:imax,jmin:jmax) :: vw
```

```
Real(rp), INTENT(INOUT), Dimension(imin:imax,jmin:jmax) :: vc
```

```
Real(rp), INTENT(INOUT), Dimension(imin:imax,jmin:jmax) :: sou
```

Au lieu de :

```
Real(rp), INTENT(INOUT), Dimension(:,:) :: ve, vn, vs, vw, vc, sou
```

```
Real(rp), INTENT(IN), Dimension(:,:) :: eps
```

Exemple 2 : loopy (openmp)

```
n1:/home/piccinali/trunk/debug/datarace $ make  
gfortran -fopenmp -c loopy.F90  
gfortran -fopenmp -o f loopy.o  
GNU fortran executable ready
```

Executez le programme :

→ **export OMP_NUM_THREADS=1 ; ./f**

Notez les valeurs de sum1 et sum2

Executez avec un nombre de threads > 1 et comparez les resultats.

Utilisez un debugger pour examiner les threads.

Exemple 2 : loopy (solution)

```
n1:/home/piccinali/trunk/debug/datarace $ export OMP_NUM_THREADS=1 ; ./f
threadid= 0/ 1
sum1=      -28
      1      -4      -9      -16
sum2=      -12
      1      -1      -4      -8
```

```
n1:/home/piccinali/trunk/debug/datarace $ export OMP_NUM_THREADS=2 ; ./f
threadid= 1/ 2
threadid= 0/ 2
sum1=      -28
      1      -4      -9      -16
sum2=      -5
      1      -1      -4      -1
```

```
n1:/home/piccinali/trunk/debug/datarace $ export OMP_NUM_THREADS=6 ; ./f
threadid= 5/ 6
threadid= 4/ 6
threadid= 0/ 6
threadid= 2/ 6
threadid= 1/ 6
threadid= 3/ 6
sum1=      -28
      1      -4      -9      -16
sum2=      -5
      1      -1      -4      -1
```

Exemple 2 : loopy (solution)

Les résultats de la boucle 2 dépendent de la façon dont elle est parallélisée ou non.

Vous devriez avoir remarqué des résultats différents en fonction du nombre de threads, ce qui est typique des data race. La boucle2 n'est pas parallélisable à cause de la dépendance entre les données.

Il n'y a pas vraiment de solution, à moins de restructurer la boucle ou d'utiliser la clause reduction pour faire la somme. La plupart du temps, les résultats ne sont pas reproductibles, ce qui rend le debuggage encore plus difficile.

<http://en.wikipedia.org/wiki/OpenMP>

Exemple 3 : attach to gdb (mpi)

```

/bin/bash 106x23
n1:~/trunk/debug/intro/attach > t sbatch.sh ./f 2 2 1
n1:/home/piccinali/trunk/debug/intro/attach $ sbatch.sh ./f 2 2 1
+ export OMP_NUM_THREADS=1
+ OMP_NUM_THREADS=1
+ /usr/bin/time -p /softs/openmpi-1.4.3/bin/mpixec -n 2 -npnode 2 -x OMP_NUM_THREADS -hostfile /softs/o
penmpi-1.4.3/h ./f

/bin/bash 74x23
Tasks: 177 total, 1 running, 176 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.1%sy, 0.0%ni, 99.9%id, 0.0%wa, 0.0%hi, 0.0%si, 0.
Mem: 49554172k total, 21549604k used, 28004568k free, 444024k buffers
Swap: 50319356k total, 0k used, 50319356k free, 18599728k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 20269 piccinal  20   0 19352 1380  976 R   1  0.0   0:00.02 top -u piccin
 19322 piccinal  20   0 107m 2416 1280 S   0  0.0   0:00.03 sshd: piccina
 19333 piccinal  20   0 107m 2420 1284 S   0  0.0   0:00.05 sshd: piccina
 19334 piccinal  20   0 107m 2424 1288 S   0  0.0   0:00.03 sshd: piccina
 19335 piccinal  20   0 27380 9.9m 1712 S   0  0.0   0:00.41 -bash
 19338 piccinal  20   0 27364 9.8m 1704 S   0  0.0   0:00.39 -bash
 19343 piccinal  20   0 27364 9.8m 1704 S   0  0.0   0:00.40 -bash
 20241 piccinal  20   0 11108 1692 1224 S   0  0.0   0:00.00 /bin/bash /ho
 20247 piccinal  20   0 11104 1684 1224 S   0  0.0   0:00.00 /bin/bash /ho
 20259 piccinal  20   0 3988 324 252 S   0  0.0   0:00.00 /usr/bin/time
 20260 piccinal  20   0 53488 2356 1708 S   0  0.0   0:00.01 /softs/openmp
 20261 piccinal  20   0 182m 5984 2632 S   0  0.0   0:00.02 ./f
 20262 piccinal  20   0 183m 5992 2640 S   0  0.0   0:00.02 ./f

n1:~/trunk/debug/intro/attach > top -u $USER

/bin/bash 161x25
n1:~/trunk/debug/intro/attach > mpif90 -g -D_MPI mpiomp.F90 -L/softs/affinity/ -laff -o f
n1:~/trunk/debug/intro/attach > ls
f mpiomp.F90 readme.jg
n1:~/trunk/debug/intro/attach > gdb --pid=20262 ./f
GNU gdb (Ubuntu/Linaro 7.2-1ubuntu1) 7.2
Copyright (C) 2010 Free Software Foundation, Inc.
```

Exemple 3 : attach to gdb (mpi)

```

/bin/bash 106x10
+ exit 0
n1:~/trunk/debug/intro/attach >

/bin/bash 160x37
n1:~/trunk/debug/intro/attach > touch go
n1:~/trunk/debug/intro/attach >

(gdb) bt
#0 0x00002b05c72135ad in nanosleep () at ../sysdeps/unix/syscall-template.S:82
#1 0x00002b05c721343c in __sleep (seconds=0) at ../sysdeps/unix/sysv/linux/sleep.c:138
#2 0x0000000000400f2c in who () at mpiomp.F90:27
#3 0x00000000004010c9 in main (argc=1, argv=0x7fff59cf41c0 "./f") at mpiomp.F90:3
#4 0x00002b05c7184eff in __libc_start_main (main=0x401090 <main>, argc=1, ubp_av=0x7fff59cf2ad8, init=<value optimized out>, fini=<
    rtd_fini=<value optimized out>, stack_end=0x7fff59cf2ac8) at libc-start.c:226
#5 0x0000000000400d99 in _start ()
(gdb) break mpiomp.F90:32
Breakpoint 1 at 0x400f94: file mpiomp.F90, line 32.
(gdb) cont
Continuing.

Breakpoint 1, who () at mpiomp.F90:32
32          rank," /",nb_procs,&
(gdb) print rank
$1 = 1
(gdb) print nb_procs
$2 = 2
(gdb) q
A debugging session is active.

        Inferior 1 [process 20262] will be detached.

Quit anyway? (y or n) y
Detaching from program: /home/piccinali/trunk/debug/intro/attach/f, process 20262
n1:~/trunk/debug/intro/attach >

```