



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS

Swiss National Supercomputing Centre



Debugging Tools

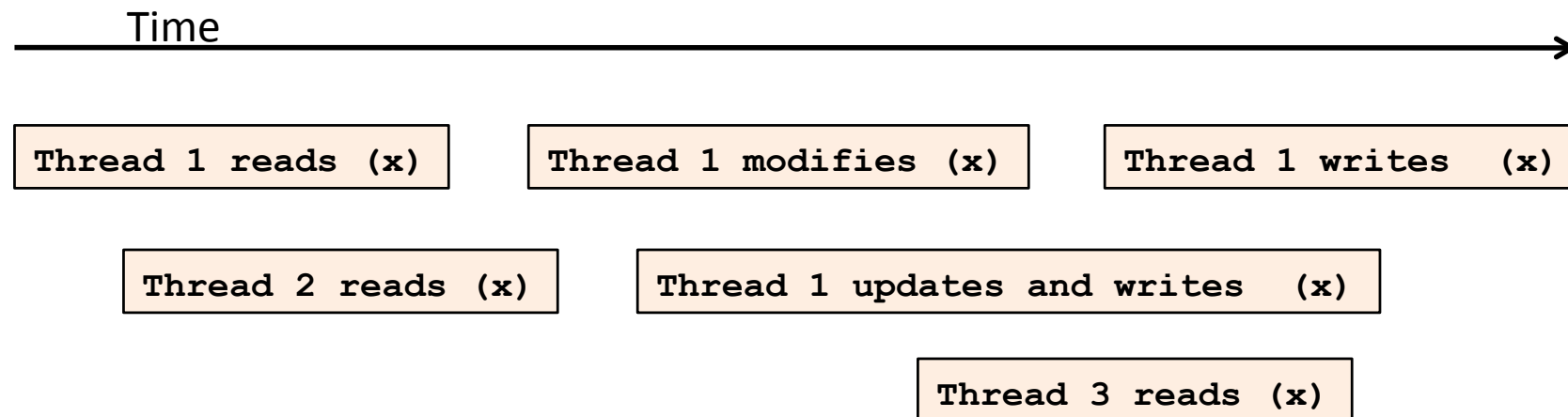
Multi-threaded Programming, Tuning and
Optimization on Multi-core MPP Platforms

February 15-17, 2011

CSCS Manno

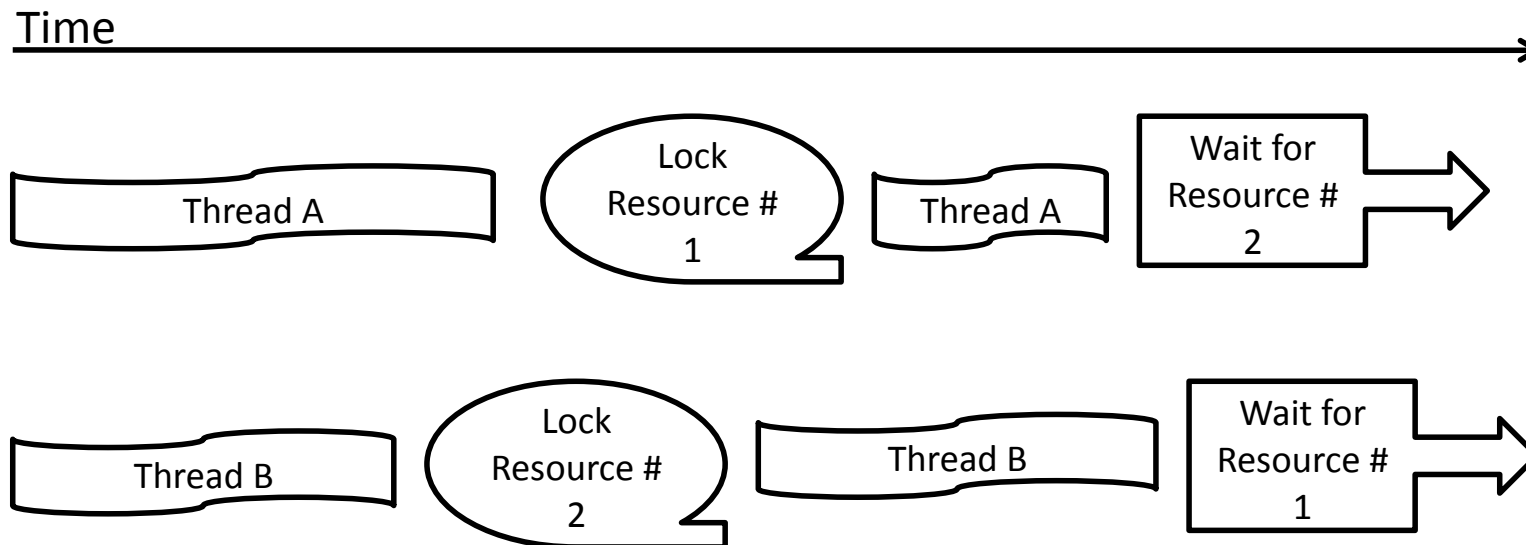
Potential Errors in Multithreaded Codes (1)

- Data Race Condition
 - Two or more threads accessing and updating same memory location
 - Unintended and unsynchronized access to shared variable resulting in unexpected behavior



Potential Errors in Multithreaded Codes (2)

- Deadlocks
 - Threads waiting on a resource that never becomes available



Deadlock Examples

```
!$OMP PARALLEL DEFAULT(SHARED)

!$OMP CRITICAL
    DO I = 1, 10
        X= X + 1
!$OMP    BARRIER
        Y= Y + I*I
    END DO
!$OMP END CRITICAL

!$OMP END PARALLEL
```

```
#pragma omp parallel
    myid = omp_get_thread_num();
    if (myid %2) {
        // do some odd work
        #pragma omp barrier
        // do more work
    }
    else {
        // do some even work
        #pragma omp barrier
        // do more work
    }
}
```

How To Avoid Deadlocks

- Using Barrier in for selected number of threads
- Avoid the lock functions as much as possible
- Avoid nesting of locks



Potential Errors in Multithreaded Codes (3)

- Livelock
 - Infinite loop conditions for certain loops (traditional loop errors)
- Memory issues
 - Stack overflows due to replication of private data items
 - Memory leaks

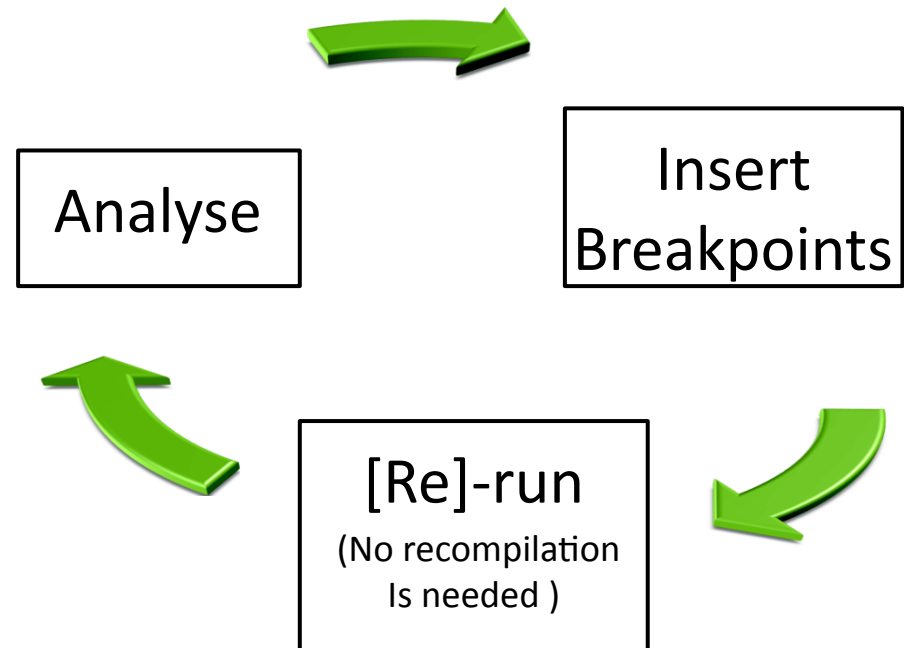


Debugging

What is going on inside your application ?

A **debugger** will follow the program through execution so you can watch your program execute step by step and view the contents of memory :

- * Let you examine the program execution **step by step**
- * Make the program **stops on specified places** or on specified conditions
- * Give information about current **variables' values**, the memory and stack



Important Debugging Concepts

- Stopping and watching a program execution at a certain point
 - Breakpoints
 - Watchpoints
- Stepping and continuing to control program execution
 - Single step lines of code
 - Single step assembler instructions
 - Resume program execution
- Examining the stack
 - Backtracing



Debugging Considerations Unique to OpenMP

- Challenges
 - Nondeterministic failures
 - Reproducibility and predictability of bugs
 - Compiler transformation manifests complex bugs
 - Memory analysis for data scoping constructs
- Use multi-threading feature of a debugger to:
 - Identify exactly when the failure occur (thread level)
 - Review program execution and memory contents
 - Inspect core files after a crash



Debugging Tools Available on CSCS Platforms

- Two tools available at the moment
 - Totalview (for hybrid MPI+OpenMP applications)
 - `module avail xt-totalview`
 - gdb
 - `module avail gdb`
- Must compile with the `-g` compiler flag
- For failures or crashes that occur after long execution times
 - Cray Fast Track Debugging (compile with `-gFast`)—only available for Cray compilers
 - For further info: <http://docs.cray.com/books/S-9401-0909//S-9401-0909.pdf>



Other Tools to Aid Multi-threaded Programming

- Debugging tools
 - Allinea DDT (MPI + OpenMP)
 - Debugger from different compiler vendors
 - ...
- Correctness tools
 - Intel Inspector
 - Rogue Wave ThreadSpotter
 - ...



Further Info

- Debugging and Performance Analysis Tools available at CSCS <http://www.cscs.ch/145.0.html>
- Craydocs <http://docs.cray.com/>
- Totalview tools (deubgger, memory scape and replay engine) <http://www.totalviewtech.com/home/>
- Allinea DDT <http://www.allinea.com/products/ddt/>
- Acumem ThreadSpotter and SlowSpotter <http://www.acumem.com>
- Eclipse Parallel Tools Platform <http://www.eclipse.org/ptp/>
- Intel Parallel Inspector <http://software.intel.com/en-us/articles/intel-parallel-inspector/>

Totalview

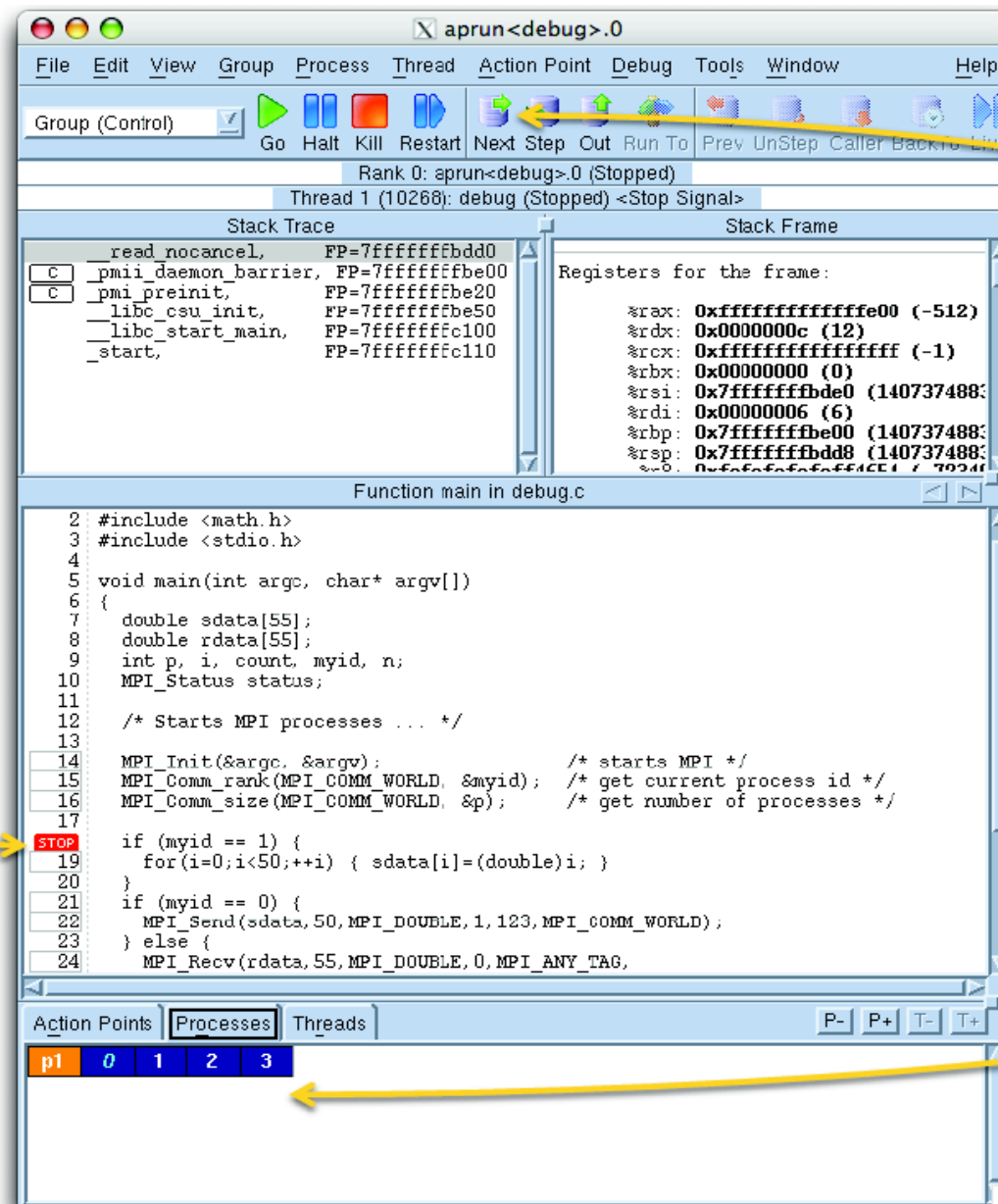
- TotalView is a debugger with support for Fortran, C, C++, MPI, OpenMP and threads.
- TotalView is an interactive tool that lets you debug serial, multiprocessor and multithreaded programs. It can be executed either as a graphical user interface (by using the totalview executable) or from a command-line interface (by using the totalviewcli executable). Totalview provides source-level debugging of Fortran and Fortran 90, C, and C++ codes. It can be used to debug parallel programs based on MPI. It also has facilities for multi-process thread-based parallel programs such as OpenMP.

```
palu1: module load PrgEnv-gnu
palu1: module list
Currently Loaded Modulefiles:
 1) modules/3.2.6.6
 2) nodestat/2.2-1.0301.24557.5.3.gem
 3) sdb/1.0-1.0301.24568.5.18.gem
 4) MySQL/5.0.64-1.0301.2899.20.1.gem
 5) lustre-cray_gem_s/1.8.2_2.6.27.48_0.12.1_1.0301.
 6) udreg/2.1-1.0301.2797.5.2.gem
 7) ugni/2.1-1.0301.2798.5.2.gem
 8) gni-headers/2.1-1.0301.2792.5.1.gem
 9) dmapp/2.2-1.0301.2791.5.1.gem
10) xpmem/0.1-2.0301.24575.5.2.gem
11) slurm
12) Base-opts/1.0.2-1.0301.24518.5.1.gem
13) xtpe-network-gemini
14) gcc/4.5.1
15) totalview-support/1.1.1
16) xt-totalview/8.8.0a
17) xt-libsci/10.4.9
18) xt-mpt/5.1.2
19) pmi/1.0-1.0000.8160.39.2.gem
20) /opt/cray/xt-asyncpe/4.5/modulefiles/xtpe-mc12
21) xt-asyncpe/4.5
22) PrgEnv-gnu/3.1.49A
palu1: █
```



Breakpoints

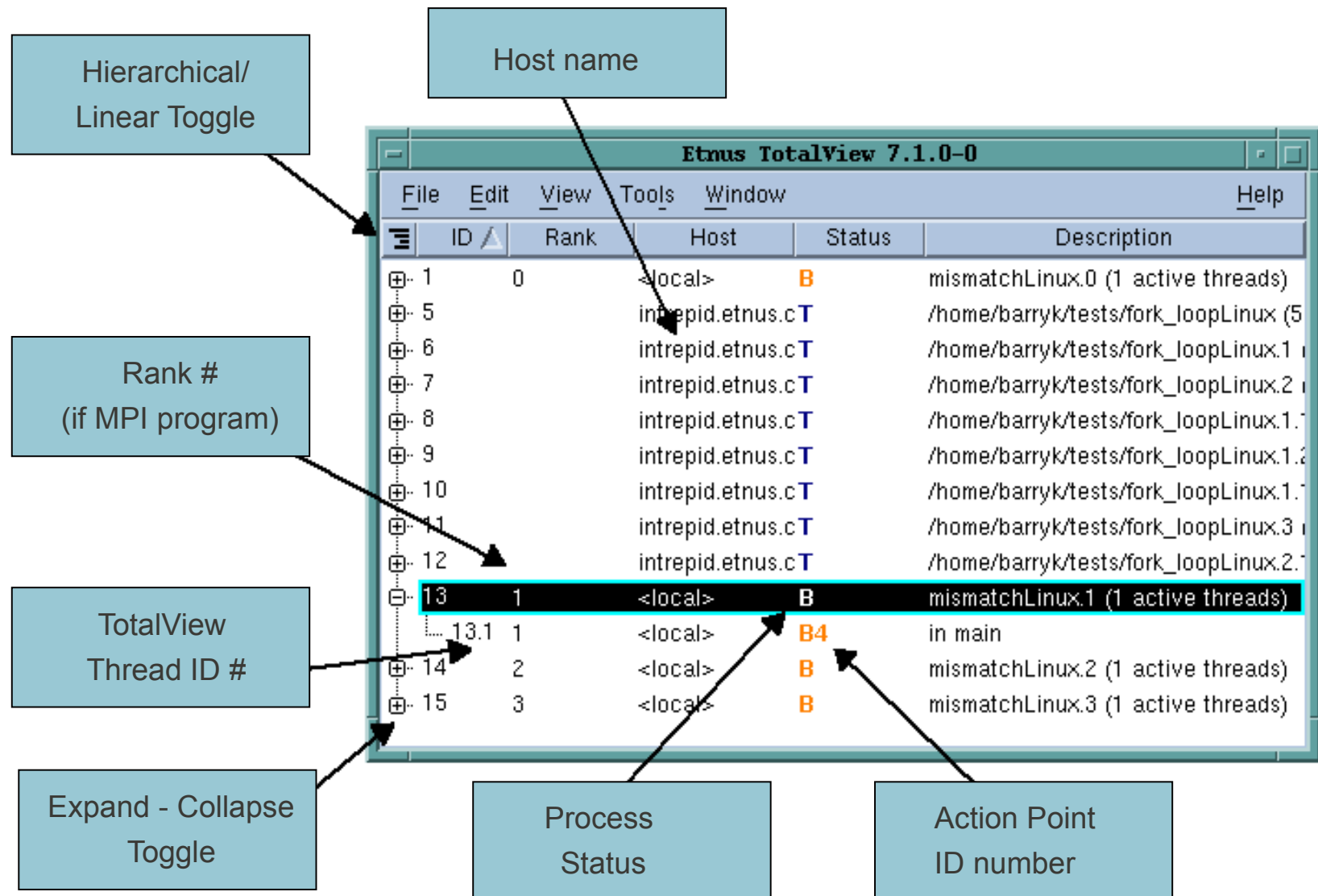
Set
Breakpoint



Execution
Toolbars

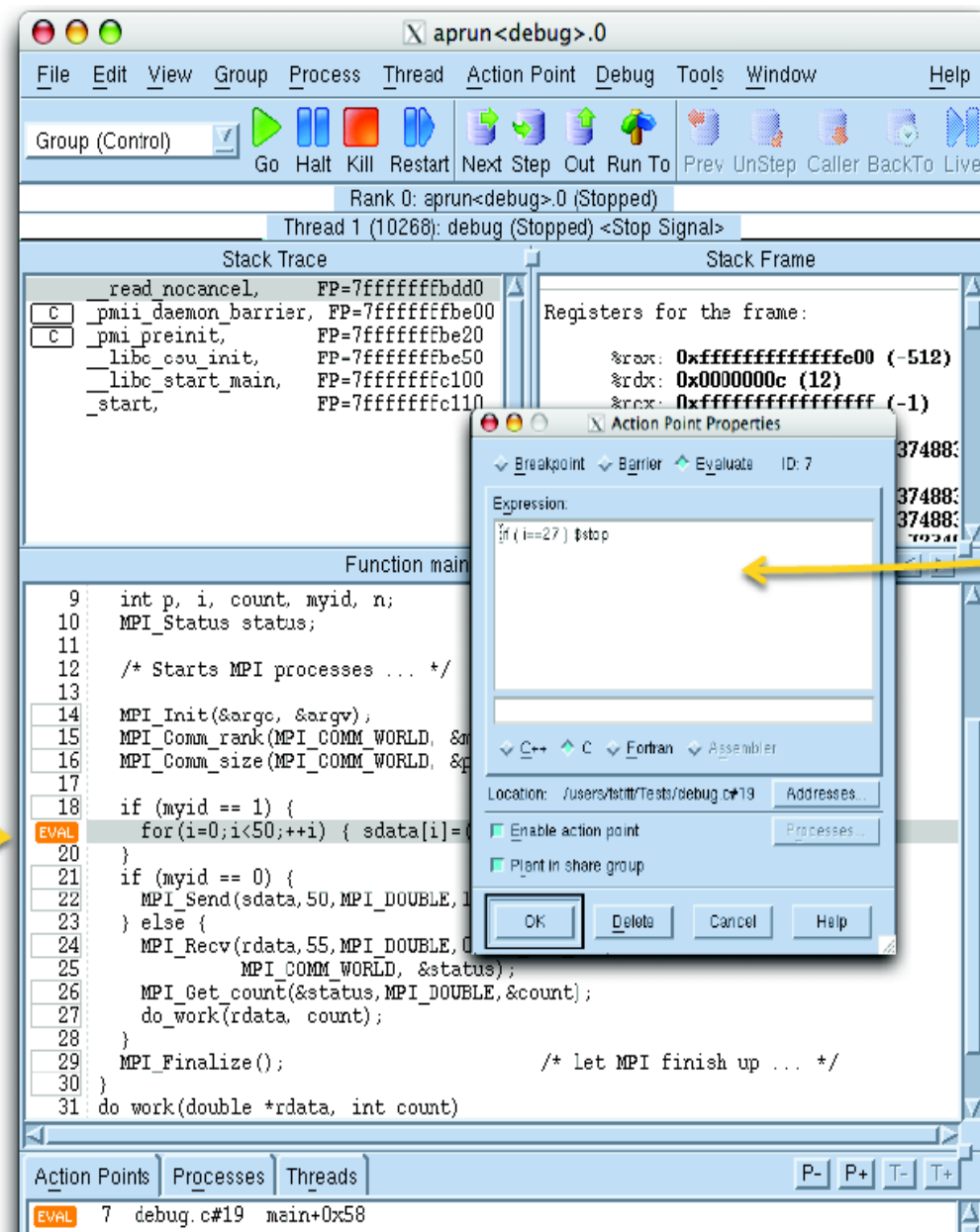
Co-operating
Processes

Root window



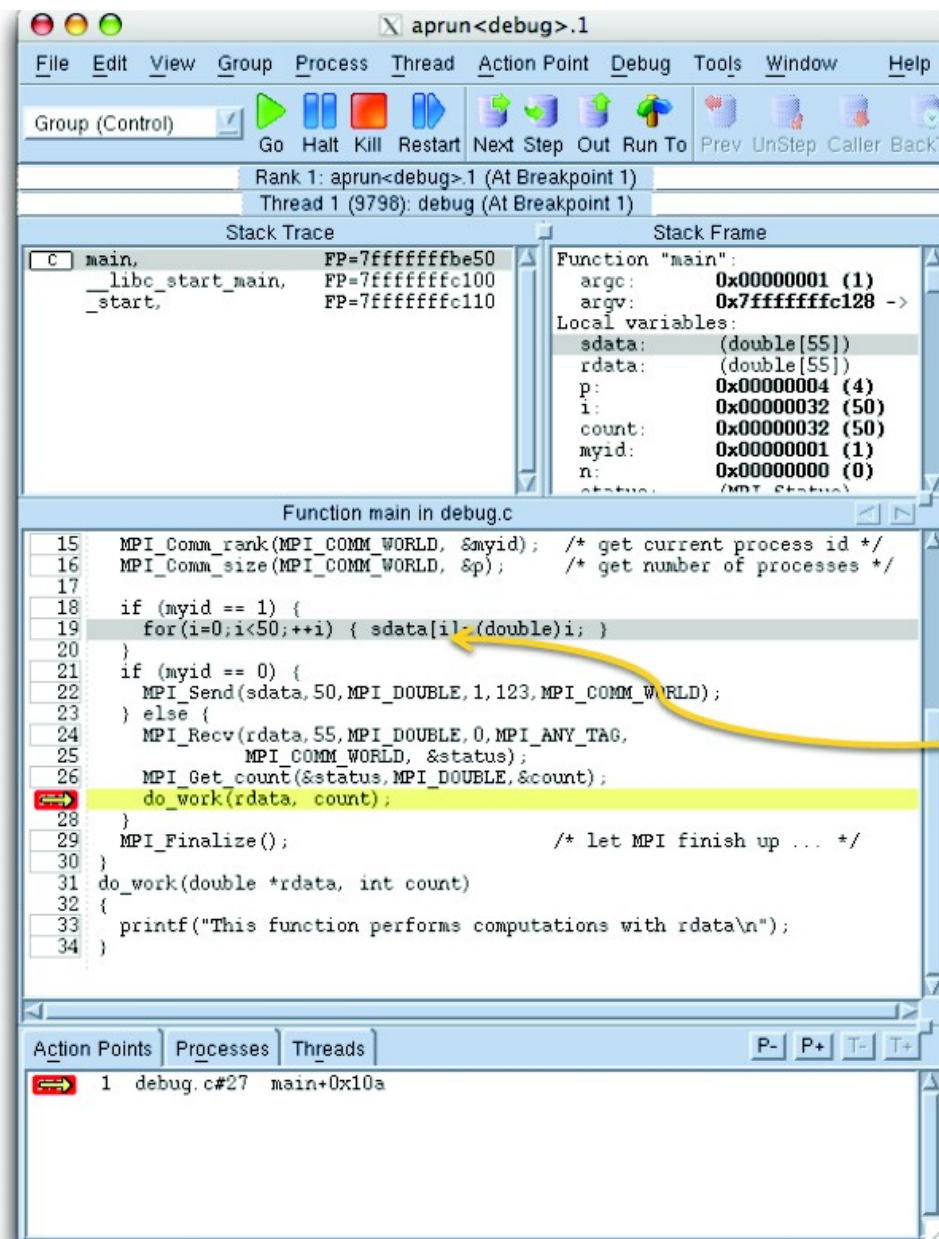
Action points

Set
Evaluation Point



Enter
Stopping
Condition

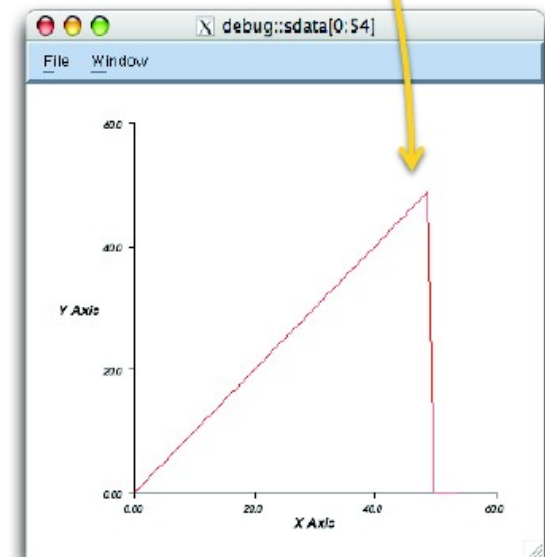
Examining data



Dive in
And
Visualize

The screenshot shows the sdata - main - 3.1 window. The top menu bar includes File, Edit, View, Tools, Window, and Help. Below the menu is a toolbar with buttons for Expression, Address, Slice, Filter, and Type. The main window shows a table of the sdata array. The table has two columns: Field and Value. The data is as follows:

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

aprun<debug>.0

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

Group (Control) Go Halt Kill Restart Next Step Out Run To Pre

Rank 0: aprun<debug>.0 (At Breakpoint 2) [M]

Thread 1 (9191): debug (At Breakpoint 2)

Stack Trace

Function	FP
main,	FP=7fffffffcd0
libc_start_main,	FP=7fffffffcc480
_start,	FP=7fffffffcc490

Function "main"

argc:

argv:

Local variables

sdata:

rdata:

p:

i:

count:

myid:

n:

status:

Function main in debug.c

```
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); /* starts MPI */
15  MPI_Comm_rank(MPI_COMM_WORLD, &myid); /* get current process id */
16  MPI_Comm_size(MPI_COMM_WORLD, &p); /* get number of processes */
17
18  if (myid == 0) {
19    for(i=0; i<50; ++i) { sdata[i]=(double)i; }
20  }
21  if (myid == 0) {
22    MPI_Send(sdata, 50, MPI_DOUBLE, 1, 123, MPI_COMM_WORLD);
23  } else {
24    MPI_Recv(rdata, 55, MPI_DOUBLE, 0, MPI_ANY_TAG,
25            MPI_COMM_WORLD, &status);
26    MPI_Get_count(&status, MPI_DOUBLE, &count);
27    do_work(rdata, count);
28  }
29 }
```

myid - main - 2.1

File Edit View Tools Window Help

2.1

Expression: myid Address: 0x7ffffffbe44

Slice:

Filter:

Type: int

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

Dive Enter

Dive In New Window

Dive In All

Expand All Ctrl++

Collapse All Ctrl+-

Undive

Undive All

Recurse

Recurse All

Freeze

Lock Address

Show Across

Compilation Scope

Loader Symbols

Padding

Break At Newlines

Examine Format

Stack Status

None Ctrl+0

Processes Ctrl+Shift+L

Threads Ctrl+L

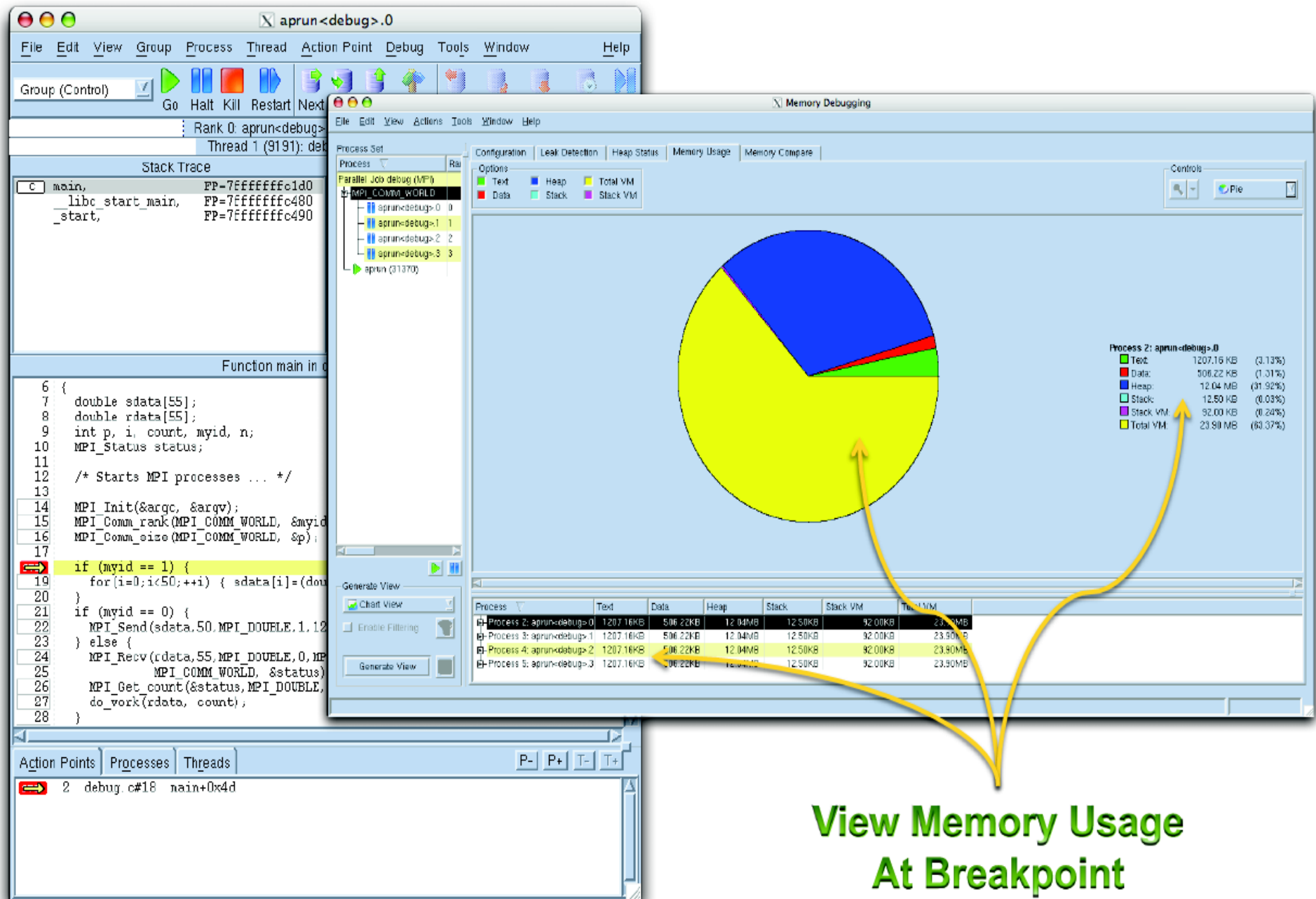
Action Points Processes Threads

P- P+ T- T+

2 debug.c#18 main+0x4d

Dive into myid

Memory usage



Memory leaks

The screenshot displays the `aprun <debug>.0` interface, which is used for debugging MPI applications. The interface is divided into several panes:

- Process List:** Shows the process set, including `Parallel Job debug (MPI)` and `MPI_COMM_WORLD`.
- Stack Trace:** Displays the current stack trace, showing the main function and its subroutines.
- Code Editor:** Shows the source code of the application. A breakpoint is set at line 19, which is highlighted in yellow.
- Memory Debugging Panel:** This panel is used to analyze memory usage and leaks. It includes a **Configuration** tab, a **Leak Detection** tab, and a **Heap Status** tab. The **Leak Detection** tab is currently selected, showing a large green memory dump. Below the dump, there is a table of **Heap Information** and a table of **Related Blocks**.

The **Heap Information** table shows the following data:

Category	Bytes	Count
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	64
Filtered	0	0
Unfiltered	15.56KB	64

The **Related Blocks** table is currently empty. A yellow arrow points from the **Leaked** row in the **Heap Information** table to the **Leaked Block** section of the **Memory Debugging** panel.

View Memory Leaks At Breakpoint

Demo : Totalview



Getting started (1)

```
/bin/bash 60x38
1      program affinity
2      use omp_lib
3
4      #ifdef _MPI
5          use mpi
6      #endif
7          IMPLICIT NONE
8
9          integer, external :: running_on
10         character(len=8) :: nid
11         integer :: coreid
12         integer :: rc=0, rank=0, numtask=0, corespersn=24
13         integer :: threadid=0, nthreads=0
14
15     #ifdef _MPI
16         call MPI_INIT ( rc )
17         call MPI_COMM_RANK ( MPI_COMM_WORLD, rank, rc )
18         call MPI_COMM_SIZE ( MPI_COMM_WORLD, numtask, rc )
19     #endif
20
21     !$omp parallel private(threadid,coreid,nid)
22     !$         threadid = omp_get_thread_num()
23     !$         nthreads = omp_get_num_threads()
24
25         call print_nodeaffinity(nid)
26         coreid = running_on()
27         print *, &
28             "rank=",rank,"/",numtask,
29             " cnid=",trim(nid(1:8)),
30             " threadid=",threadid,"/",nthreads,
31             " core=", coreid
32     !$omp end parallel
33     #ifdef _MPI
34         call MPI_FINALIZE ( rc )
35     #endif
36
37     end program affinity
"aff3.F90" 37 lines --2%--
```


Getting started (2)

```
palu1: ls
aff2.F90 Makefile runningon.c
palu1: make FFLAGS="-g -fopenmp -D_MPI"
cc -g -fopenmp -D_MPI -c runningon.c
ftn -g -fopenmp -D_MPI -c aff2.F90
ftn -g -fopenmp -D_MPI -o exe runningon.o aff2.o
/usr/lib/../../lib64/libpthread.a(sem_open.o): In function `sem_open':
/usr/src/packages/BUILD/glibc-2.9/nptl/sem_open.c:330: warning: the use of `mktemp' is dangerous, better use `mkstemp'
palu1:
palu1:
palu1: salloc -N2
salloc: Granted job allocation 2959
palu1: queue -u $USER
```

JOBID	USER	ACCOUNT	NAME	PARTITION	ST	EXEC_HOST	REASON	START_TIME	TIME	TIME_LEFT	NODES	PRIORITY
2959	piccinal	csstaff	bash	day	R	palu1	None	2011-02-09T09:58	0:08	59:52	2	2

```
palu1:
palu1: export OMP_NUM_THREADS=2 ; aprun -n2 -N1 -d24 exe
rank=      0 /      2 cnid=nid00003 threadid=      1 /      2 core=      1
rank=      0 /      2 cnid=nid00003 threadid=      0 /      2 core=      0
rank=      1 /      2 cnid=nid00004 threadid=      0 /      2 core=      0
rank=      1 /      2 cnid=nid00004 threadid=      1 /      2 core=      1
done
done
Application 108245 resources: utime ~0s, stime ~0s
palu1: exit
exit
salloc: Relinquishing job allocation 2959
salloc: Job allocation 2959 has been revoked.
palu1: █
```

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 0xff0000009009d800
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 0xff0000009009e100
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 0xff000000900a0500
Mapping 14508 bytes of ELF string data from '/lib64/libpthr
Indexing 12804 bytes of DWARF '.eh_frame' symbols from '/lib
Library /lib64/libc.so.6, with 2 asects, was linked at 0x00
at 0xff000000900baa00
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 0xff0000009007ee00
Mapping 380 bytes of ELF string data from '/lib64/ld-linux-x
Indexing 6072 bytes of DWARF '.eh_frame' symbols from '/lib6
one
  
```

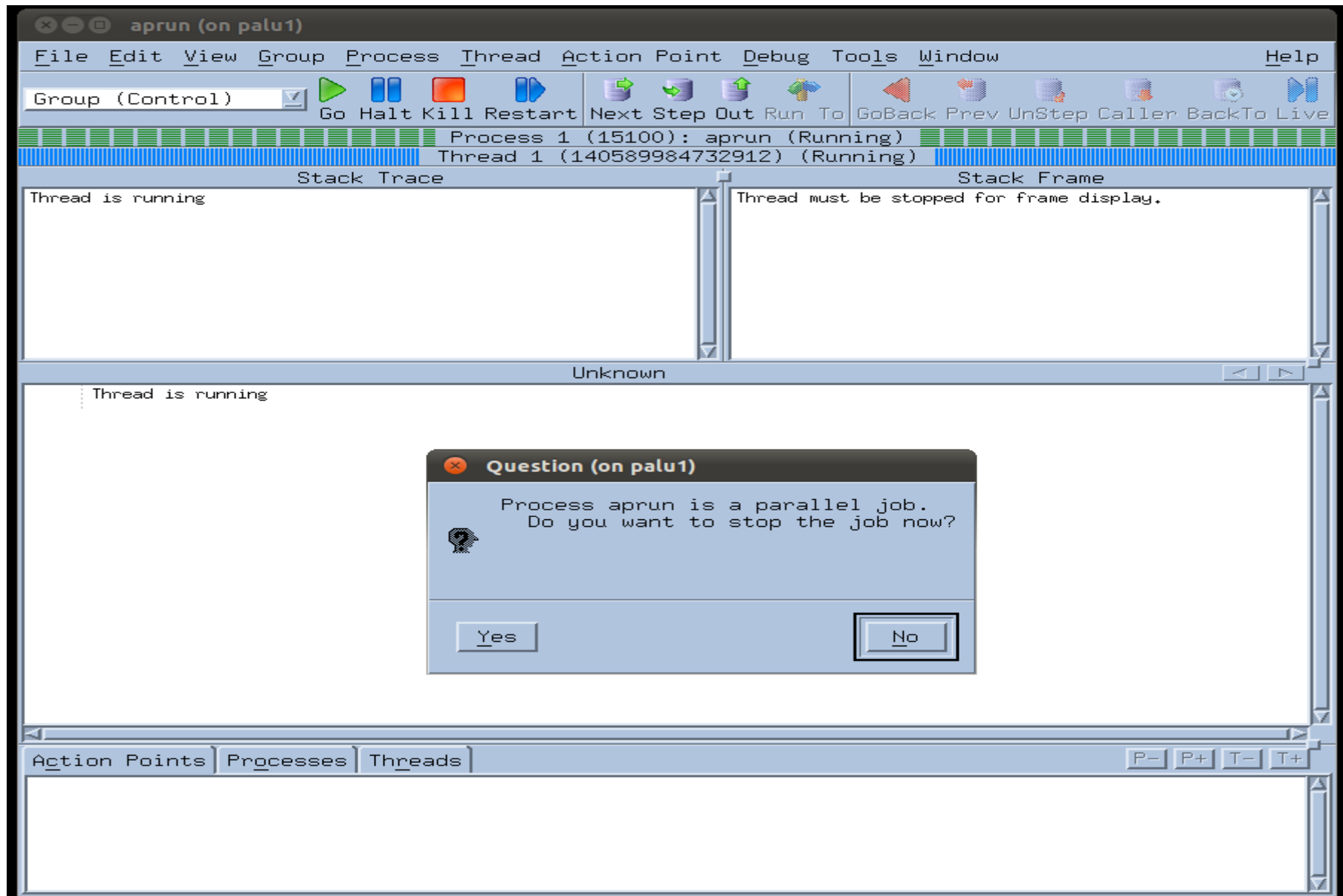
File	Edit	View	Tools	Window	Help
ID	Rank	Host	Status	Description	
1		<local>	-	aprun (0 active threads)	

The screenshot shows the TotalView GUI with the following components:

- Root window:** Displays process and thread information. It shows a table with columns: ID, Rank, Host, Status, and Description. The table contains one entry: 1, <local>, -, aprun (0 active threads).
- Process window:** Displays stack trace, stack frame, and source code for the selected thread in a process. It shows a stack trace for Process 1 (0): aprun (Exited or Never Created). The stack trace is empty, showing "No current thread".
- Variable window:** Displays address, data type, and value of local variable, register, or global variable. It shows a message: "Parallel program has not yet been started."

Root window	Lists 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

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, _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 Editor:** 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
- Thread List:**

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 __ioct1
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 __ioct1
3.3		1 nid00004	T	in MAIN__omp_fn.0
- Action Points Panel:**

Action Points	Processes	Threads
2.1	(5614) B1	in MAIN__omp_fn.0
2.2	(5659) T	in __ioct1
2.3	(5660) T	in MAIN__omp_fn.0

Viewing data across processes and threads

aprun<exe>.0 (on palu1)

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: aprun<exe>.0 (At Breakpoint 2)
Thread 1 (5732): exe (At Breakpoint 2)

Stack Trace

f90 affinity.
f90 main.
C __libc_start_main.
_start,

Stack Frame

Function "affinity":
No arguments.
Local variables:
coreid: 0 (0x00000000)
nid: ""
nthreads: 2 (0x00000002)
numtask: 2 (0x00000002)
rank: 0 (0x00000000)
rc: 0 (0x00000000)
threadid: 0 (0x00000000)
Modules:

Function affinity in aff2.F90

```

30  !$  nthreads = omp_get_num_threads()
31  !!  corespعرن = 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

```

Action Points

p1 0 1

Dive
Add to Expression List
Across Processes
Across Threads
Set Breakpoint
Set Barrier
Create Watchpoint
Enable
Disable
Delete
Properties

File Edit View Tools Window Help

6.1

Expression: rank Address: 0x007b1e94
Slice: Filter:
Type: integer(kind=4)

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

File Edit View Tools Window Help

6.1

Expression: threadid Address: 0x007b1e9c
Slice: Filter:
Type: integer(kind=4)

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 0xffffffff700000, 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:
svr:launch /opt/toolworks/totalview.8.8.0a/linux-x86-64/bin/tvdsrvrmain '-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: █
```

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[ele]te [<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



Getting started

```
/bin/bash 139x35
palu2: module load PrgEnv-gnu
palu2: module load gdb
palu2: module list
Currently Loaded Modulefiles:
 1) modules/3.2.6.6
 2) nodestat/2.2-1.0301.24557.5.3.gem
 3) sdb/1.0-1.0301.24568.5.18.gem
 4) MySQL/5.0.64-1.0301.2899.20.1.gem
 5) lustre-cray_gem_s/1.8.2.2.6.27.48_0.12.1_1.0301.5636.4.1-1.0301.24584.3.22
 6) udreg/2.1-1.0301.2797.5.2.gem
 7) ugni/2.1-1.0301.2798.5.2.gem
 8) gni-headers/2.1-1.0301.2792.5.1.gem
 9) dmapp/2.2-1.0301.2791.5.1.gem
10) xpmem/0.1-2.0301.24575.5.2.gem
11) slurm
12) Base-opts/1.0.2-1.0301.24518.5.1.gem
13) xtpe-network-gemini
14) gcc/4.5.1
15) totalview-support/1.1.1
16) xt-totalview/8.8.0a
17) xt-libsci/10.4.9
18) xt-mpt/5.1.2
19) pmi/1.0-1.0000.8160.39.2.gem
20) /opt/cray/xt-asyncpe/4.5/modulefiles/xtpe-mc12
21) xt-asyncpe/4.5
22) PrgEnv-gnu/3.1.49A
23) gdb/7.2
palu2: make clean ; make FFLAGS="-g -fopenmp" ; mv exe omp
rm -f exe *.o *.mod
cc -g -fopenmp -c runningon.c
ftn -g -fopenmp -c aff2.F90
ftn -g -fopenmp -o exe runningon.o aff2.o
/usr/lib/./lib64/libpthread.a(sem_open.o): In function `sem_open':
/usr/src/packages/BUILD/glibc-2.9/nptl/sem_open.c:330: warning: the use of `mktemp' is dangerous, better use `mkstemp'
palu2:

/bin/bash 51x41
22
23 !$omp parallel private(threadid,coreid,nid)
24 !$   threadid = omp_get_thread_num()
25 !$   nthreads = omp_get_num_threads()
26 !!   corespncr = omp_get_num_procs()
27 !!   MAXT = OMP_GET_MAX_THREADS()
28 !!   INPAR = OMP_IN_PARALLEL()
29 !!   DYNAMIC = OMP_GET_DYNAMIC()
30 !!   NESTED = OMP_GET_NESTED()
31
32   call print_nodeaffinity(nid)
33 !   call print_coreaffinity(coreid)
34   coreid = running_on()
35 ! barrier
36   print *, &
37     "rank=",rank,"/",numtask,
38     " cnid=",trim(nid(1:8)),
39     " threadid=",threadid,"/",nthreads,
40     " core=", coreid
41 !$omp end parallel
42 #ifdef _MPI
43   call MPI_FINALIZE ( rc )
44 #endif
45   print *, "done"
46
47   end program affinity

27,1 Bot
```

Launching GDB

```
palu2: salloc -N1
salloc: Granted job allocation 2982
palu2: export OMP_NUM_THREADS=24 ; aprun -nl -d24 ./omp
rank=      0 /      0 cnid=nid00002 threadid=      2 /      24 core=      2
rank=      0 /      0 cnid=nid00002 threadid=     18 /      24 core=     18
rank=      0 /      0 cnid=nid00002 threadid=      4 /      24 core=      4
rank=      0 /      0 cnid=nid00002 threadid=      7 /      24 core=      7
rank=      0 /      0 cnid=nid00002 threadid=     11 /      24 core=     11
rank=      0 /      0 cnid=nid00002 threadid=      8 /      24 core=      8
rank=      0 /      0 cnid=nid00002 threadid=      5 /      24 core=      5
rank=      0 /      0 cnid=nid00002 threadid=      3 /      24 core=      3
rank=      0 /      0 cnid=nid00002 threadid=      6 /      24 core=      6
rank=      0 /      0 cnid=nid00002 threadid=     13 /      24 core=     13
rank=      0 /      0 cnid=nid00002 threadid=     17 /      24 core=     17
rank=      0 /      0 cnid=nid00002 threadid=     14 /      24 core=     14
rank=      0 /      0 cnid=nid00002 threadid=      0 /      24 core=      0
rank=      0 /      0 cnid=nid00002 threadid=     15 /      24 core=     15
rank=      0 /      0 cnid=nid00002 threadid=     10 /      24 core=     10
rank=      0 /      0 cnid=nid00002 threadid=      9 /      24 core=      9
rank=      0 /      0 cnid=nid00002 threadid=     21 /      24 core=     21
rank=      0 /      0 cnid=nid00002 threadid=     23 /      24 core=     23
rank=      0 /      0 cnid=nid00002 threadid=     19 /      24 core=     19
rank=      0 /      0 cnid=nid00002 threadid=      1 /      24 core=      1
rank=      0 /      0 cnid=nid00002 threadid=     16 /      24 core=     16
rank=      0 /      0 cnid=nid00002 threadid=     12 /      24 core=     12
rank=      0 /      0 cnid=nid00002 threadid=     22 /      24 core=     22
rank=      0 /      0 cnid=nid00002 threadid=     20 /      24 core=     20
done
Application 108315 resources: utime ~0s, stime ~0s
palu2: export OMP_NUM_THREADS=3 ; aprun -nl -d24 gdb ./omp
Could not find platform independent libraries <prefix>
Could not find platform dependent libraries <exec_prefix>
Consider setting $PYTHONHOME to <prefix>[:<exec_prefix>]
'import site' failed; use -v for traceback
Traceback (most recent call last):
  File "<string>", line 32, in ?
ImportError: No module named os.path
GNU gdb (GDB) 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-unknown-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /users/piccinal/AFF/palu/GNU/omp...done.
(gdb)
```


Inserting Breakpoints

The image shows a GDB terminal window on the left and a source code editor on the right. The terminal window is titled `/bin/bash 96x31` and shows the output of the `palu2` command, which runs `gdb ./omp`. The output includes the GDB version (7.2), copyright information, and the location of the symbols file. The user has entered the command `(gdb) break 34`, and the terminal shows the breakpoint being set at address `0x4004c6` in file `aff2.F90`, line 34. The user then enters `(gdb) break 45`, and the terminal shows the breakpoint being set at address `0x4003de` in file `aff2.F90`, line 45. The user then enters `(gdb) info break`, and the terminal shows the list of breakpoints.

The source code editor is titled `/bin/bash 50x36` and shows the source code of the `affinity` program. The code is in Fortran and includes OpenMP directives. The user has inserted two breakpoints: one at line 34 (indicated by a green arrow) and one at line 45 (indicated by a green arrow). The code is as follows:

```
24 !$ threadid = omp_get_thread_num()
25 !$ nthreads = omp_get_num_threads()
26 !! corespercn = omp_get_num_procs()
27 !! MAXT = OMP_GET_MAX_THREADS()
28 !! INPAR = OMP_IN_PARALLEL()
29 !! DYNAMIC = OMP_GET_DYNAMIC()
30 !! NESTED = OMP_GET_NESTED()
31
32 call print_nodeaffinity(nid)
33 ! call print_coreaffinity(coreid)
34 coreid = running_on()
35 ! barrier
36 print *, &
37 "rank=", rank, "/", numtask,
38 " cnid=", trim(nid(1:8)),
39 " threadid=", threadid, "/", nthreads,
40 " core=", coreid
41 !$omp end parallel
42 #ifdef MPI
43 call MPI_FINALIZE ( rc )
44 #endif
45 print *, "done"
46
47 end program affinity
```

Num	Type	Disp	Enb	Address	What
1	breakpoint	keep	y	0x00000000004004c6	in MAIN_.omp_fn.0 at aff2.F90:34
2	breakpoint	keep	y	0x00000000004003de	in affinity at aff2.F90:45

Viewing data across threads (1)

```
/bin/bash 96x31

(gdb) help run
Start debugged program.  You may specify arguments to give it.
Args may include "*", or "[...]"; they are expanded using "sh".
Input and output redirection with ">", "<", or ">>" are also allowed.

With no arguments, uses arguments last specified (with "run" or "set args").
To cancel previous arguments and run with no arguments,
use "set args" without arguments.
(gdb) run
Starting program: /users/piccinal/AFF/palu/GNU/omp
[Thread debugging using libthread_db enabled]
[New Thread 0x2aaaaacac950 (LWP 13736)]
[New Thread 0x2aaaaae950 (LWP 13737)]

Breakpoint 1, MAIN__omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
34      coreid = running_on()
(gdb) info thread
3 Thread 0x2aaaaae950 (LWP 13737)  MAIN__omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
2 Thread 0x2aaaaacac950 (LWP 13736)  MAIN__omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
* 1 Thread 0x6d38c0 (LWP 13733)  MAIN__omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
(gdb) print threadid
$1 = 0
(gdb) list
34      coreid = running_on()
35      ! barrier
36      print *, &
37      "rank=",rank,"/",numtask,      &
38      " cnid=",trim(nid(1:8)),      &
39      " threadid=",threadid,"/",nthreads, &
40      " core=", coreid
41      !$omp end parallel
```

```
/bin/bash 50x36

24 !$  threadid = omp_get_thread_num()
25 !$  nthreads = omp_get_num_threads()
26 !!  corespercn = omp_get_num_procs()
27 !   MAXT = OMP_GET_MAX_THREADS()
28 !!   INPAR = OMP_IN_PARALLEL()
29 !!   DYNAMIC = OMP_GET_DYNAMIC()
30 !!   NESTED = OMP_GET_NESTED()
31
32      call print_nodeaffinity(nid)
33 !     call print_coreaffinity(coreid)
34      coreid = running_on()
35 ! barrier
36      print *, &
37      "rank=",rank,"/",numtask,
38      " cnid=",trim(nid(1:8)),
39      " threadid=",threadid,"/",nthreads,
40      " core=", coreid
41 !$omp end parallel
42 #ifdef MPI
43      call MPI_FINALIZE ( rc )
44 #endif
45      print *,"done"
46
47      end program affinity

27,1 Bot
```

Viewing data across threads (2)

```
/bin/bash 96x31
(gdb) info thread
3 Thread 0x2aaaaae950 (LWP 13737) MAIN_.omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
2 Thread 0x2aaaaacac950 (LWP 13736) MAIN_.omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
* 1 Thread 0x6d38c0 (LWP 13733) MAIN_.omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
(gdb) thread 2
[Switching to thread 2 (Thread 0x2aaaaacac950 (LWP 13736))]#0 MAIN_.omp_fn.0 (.omp_data_i=0x0)
at aff2.F90:34
34         coreid = running_on()
(gdb) info thread
3 Thread 0x2aaaaae950 (LWP 13737) MAIN_.omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
* 2 Thread 0x2aaaaacac950 (LWP 13736) MAIN_.omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
1 Thread 0x6d38c0 (LWP 13733) MAIN_.omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
(gdb) print threadid
$2 = 1
(gdb) thread 3
[Switching to thread 3 (Thread 0x2aaaaae950 (LWP 13737))]#0 MAIN_.omp_fn.0 (.omp_data_i=0x0)
at aff2.F90:34
34         coreid = running_on()
(gdb) print threadid
$3 = 2
(gdb) help cont
Continue program being debugged, after signal or breakpoint.
If proceeding from breakpoint, a number N may be used as an argument,
which means to set the ignore count of that breakpoint to N - 1 (so that
the breakpoint won't break until the Nth time it is reached).

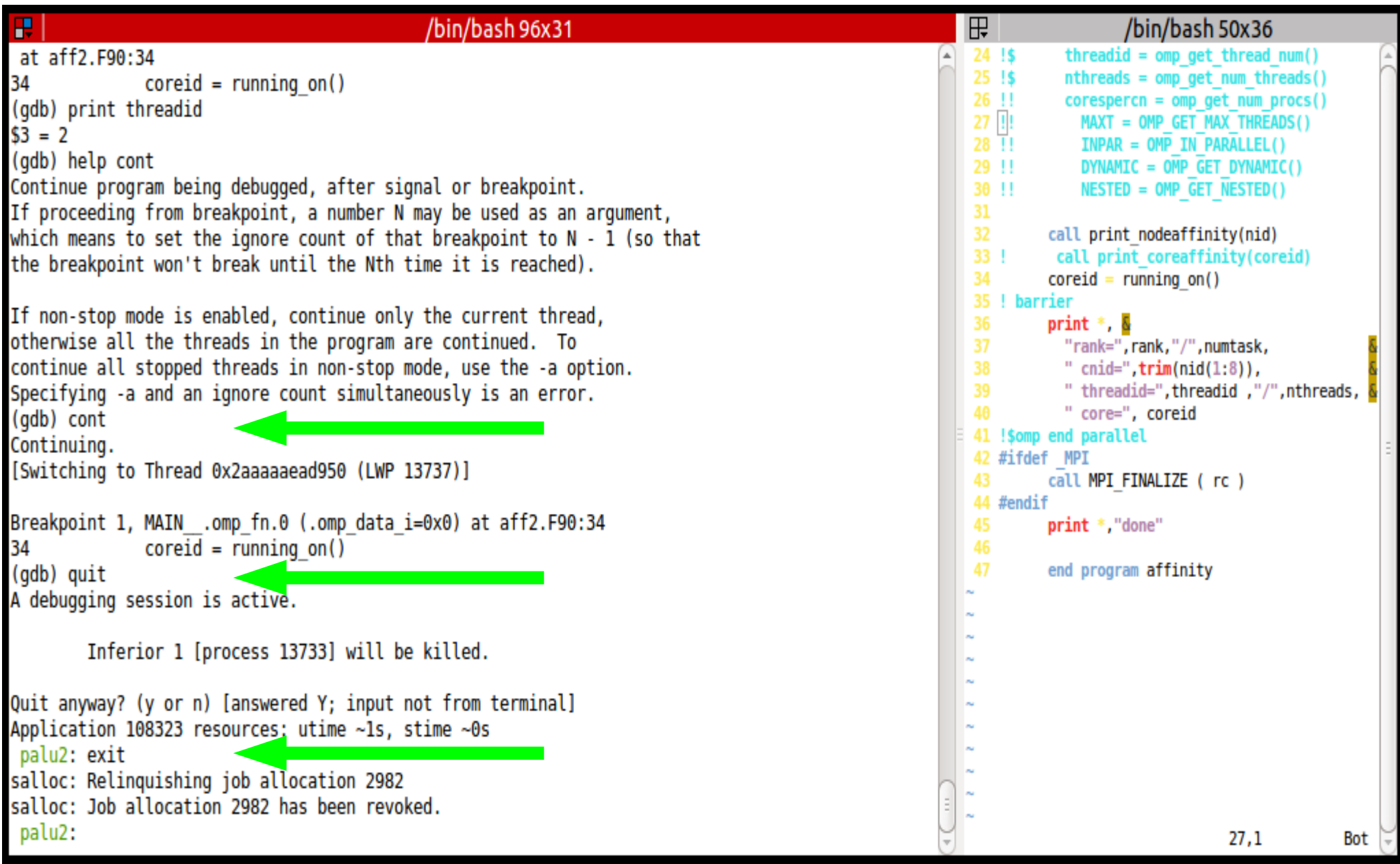
If non-stop mode is enabled, continue only the current thread,
otherwise all the threads in the program are continued. To
continue all stopped threads in non-stop mode, use the -a option.
Specifying -a and an ignore count simultaneously is an error.
(gdb)
```

```
/bin/bash 50x36
24 !$ threadid = omp_get_thread_num()
25 !$ nthreads = omp_get_num_threads()
26 !! corespercn = omp_get_num_procs()
27 !! MAXT = OMP_GET_MAX_THREADS()
28 !! INPAR = OMP_IN_PARALLEL()
29 !! DYNAMIC = OMP_GET_DYNAMIC()
30 !! NESTED = OMP_GET_NESTED()
31
32 call print_nodeaffinity(nid)
33 ! call print_coreaffinity(coreid)
34 coreid = running_on()
35 ! barrier
36 print *, &
37 "rank=",rank,"/",numtask,
38 " cnid=",trim(nid(1:8)),
39 " threadid=",threadid,"/",nthreads,
40 " core=", coreid
41 !$omp end parallel
42 #ifdef MPI
43 call MPI_FINALIZE ( rc )
44 #endif
45 print *, "done"
46
47 end program affinity
```

27,1

Bot

Restarting and exiting GDB



```
/bin/bash 96x31
at aff2.F90:34
34      coreid = running_on()
(gdb) print threadid
$3 = 2
(gdb) help cont
Continue program being debugged, after signal or breakpoint.
If proceeding from breakpoint, a number N may be used as an argument,
which means to set the ignore count of that breakpoint to N - 1 (so that
the breakpoint won't break until the Nth time it is reached).

If non-stop mode is enabled, continue only the current thread,
otherwise all the threads in the program are continued. To
continue all stopped threads in non-stop mode, use the -a option.
Specifying -a and an ignore count simultaneously is an error.
(gdb) cont
Continuing.
[Switching to Thread 0x2aaaaae9d950 (LWP 13737)]

Breakpoint 1, MAIN_.omp_fn.0 (.omp_data_i=0x0) at aff2.F90:34
34      coreid = running_on()
(gdb) quit
A debugging session is active.

        Inferior 1 [process 13733] will be killed.

Quit anyway? (y or n) [answered Y; input not from terminal]
Application 108323 resources: utime ~1s, stime ~0s
palu2: exit
salloc: Relinquishing job allocation 2982
salloc: Job allocation 2982 has been revoked.
palu2:

/bin/bash 50x36
24 !$  threadid = omp_get_thread_num()
25 !$  nthreads = omp_get_num_threads()
26 !!  corespargc = omp_get_num_procs()
27 !!  MAXT = OMP_GET_MAX_THREADS()
28 !!  INPAR = OMP_IN_PARALLEL()
29 !!  DYNAMIC = OMP_GET_DYNAMIC()
30 !!  NESTED = OMP_GET_NESTED()
31
32      call print_nodeaffinity(nid)
33 !    call print_coreaffinity(coreid)
34      coreid = running_on()
35 ! barrier
36      print *, &
37          "rank=",rank,"/",numtask,
38          " cnid=",trim(nid(1:8)),
39          " threadid=",threadid,"/",nthreads,
40          " core=", coreid
41 !$omp end parallel
42 #ifdef MPI
43      call MPI_FINALIZE ( rc )
44 #endif
45      print *, "done"
46
47      end program affinity
```


- Questions ?
- Hands-on exercises
- Thank you for your attention

Hands-on exercise1

Data race condition

- A data race occurs under the following conditions:
 - 2 or more threads in a process concurrently access the **same memory location**,
 - **At least one of the threads is** accessing the memory location for **writing**, and
 - The threads are not using any **exclusive locks** to control their accesses to that memory.
- When these three conditions hold, the order of accesses is **non-deterministic**. Therefore each run can give different results depending on the order of the accesses. Some data races may be harmless (for example, when the memory access is used for a busy-wait), but many data races are either bugs or caused by bugs in the program.

The object of this exercise is to determine whether it's safe to parallelise every DO loop that you see. Follow these steps :

- copy **loopy.f90** to your directory.
- compile the code sequentially (that is with no '-fopenmp' flag) and determine the correct result.
- parallelise every loop and run the program on 2, 6, 12 and 24 threads (you can do this interactively) and compare these results with those from above.
- What's wrong ?
- Rewrite your parallelised code to give the correct results irrespective of the number of threads used.



Hands-on exercise2

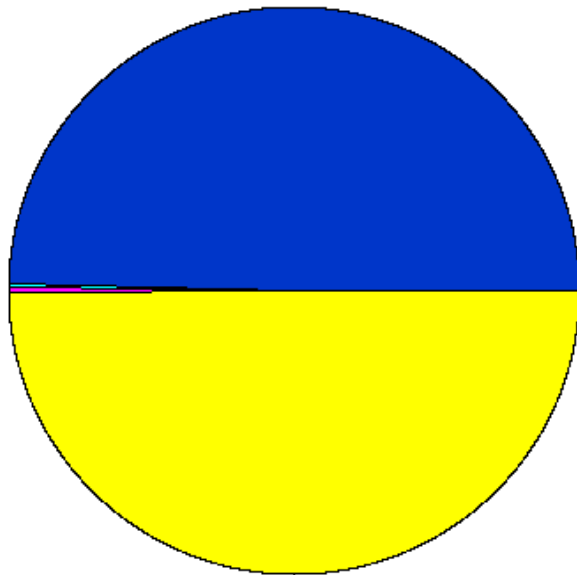
Segmentation faults

- Default **thread stack size** can be easy to exhaust. OpenMP thread stack size is an implementation dependent resource. In this case, the array is too large to fit into the thread stack space and causes the segmentation fault.
- The OpenMP standard does not specify how much stack space a thread should have. Consequently, **implementations will differ** in the default thread stack size.
- Default thread stack size can also be non-portable between compilers. Threads that exceed their stack allocation may or may not seg fault. An application may continue to run while data is being corrupted.
- **OMP_STACKSIZE** (OMP/3.0) : controls the size of the stack for (non-master) threads.
- Set the default thread stack size (in kilobytes by default) or B, K, M or G (bytes, kilobytes, megabytes or gigabytes).

The object of this exercise is to use the debuggers to find the origin of the segmentation fault. Follow these steps :

- copy **crash.f** to your directory.
- **module load PrgEnv-gnu gdb**
- compile (with '**-g -fopenmp**' flag) and run the code with any number of threads.
- What's wrong ?

Hands-on exercise2



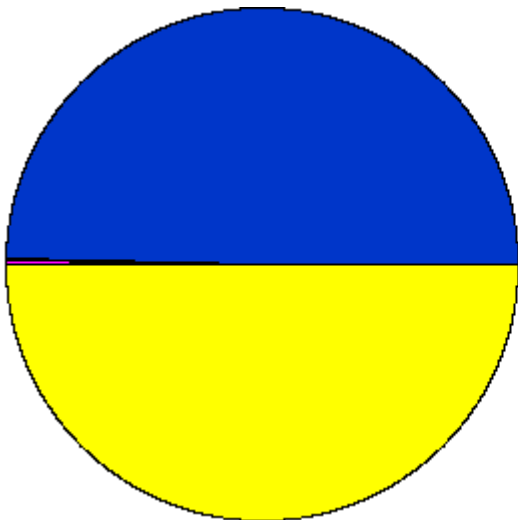
Process 2: aprun<memoryscape>.0

Text:	1881.69 KB	(0.03%)
Data:	145.23 KB	(0.00%)
Heap:	3085.24 MB	(49.53%)
Stack:	16.77 MB	(0.27%)
Stack VM:	16.78 MB	(0.27%)
Total VM:	3108.05 MB	(49.90%)



If OMP_STACKSIZE is not set, the initial value of the stacksize-var internal control variable is set to the default value.

MemoryScape only shows information for the main thread's stack.



Process 2: aprun<memoryscape>.0

Text:	1881.69 KB	(0.03%)
Data:	145.23 KB	(0.00%)
Heap:	3089.10 MB	(49.60%)
Stack:	8.39 MB	(0.13%)
Stack VM:	16.78 MB	(0.27%)
Total VM:	3111.91 MB	(49.97%)



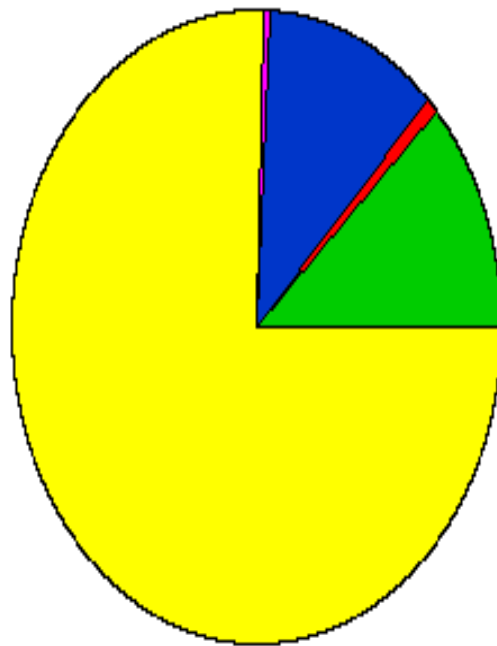
Hands-on exercise3

Deadlock

- Deadlock describes a condition where two or more threads are blocked (hang) forever, waiting for each other. Suppose we have a process with two or more threads. A deadlock occurs when the following three conditions hold :
 - Threads already holding locks request new locks,
 - The requests are made concurrently, and
 - Two or more threads form a circular chain where each thread waits for a lock that the next thread in the chain holds.
- Here is an example of a deadlock condition:
 - Thread 1: holds lock A, requests lock B
 - Thread 2: holds lock B, requests lock A
- A deadlock can be of two types: A "potential deadlock" or an "actual deadlock". A potential deadlock is a deadlock that did not occur in a given run, but can occur in different runs of the program depending on the timings of the requests for locks by the threads. An actual deadlock is one that actually occurred in a given run of the program. An actual deadlock causes the threads involved to hang, but may or may not cause the whole process to hang.

```
/bin/bash 36x11
1 !$OMP PARALLEL DEFAULT(SHARED)
2 !$OMP CRITICAL
3     DO I = 1, 10
4         X= X + 1
5 !$OMP BARRIER
6     Y= Y + I*I
7     END DO
8 !$OMP END CRITICAL
9 !$OMP END PARALLEL
~
<9L, 169C written 1,1 All
```

```
/bin/bash 37x14
1 #pragma omp parallel
2   myid = omp_get_thread_num();
3   if (myid %2) {
4       // do some odd work
5       #pragma omp barrier
6       // do more work
7   }
8   else {
9       // do some even work
10      #pragma omp barrier
11      // do more work
12  }
13 }
```



Process 1 (25994): aprun

Text:	2.33 MB	(11.82%)
Data:	167.91 KB	(0.83%)
Heap:	2.25 MB	(11.42%)
Stack:	18.30 KB	(0.09%)
Stack VM:	92.00 KB	(0.46%)
Total VM:	14.87 MB	(75.38%)

KO

Process	Text	Data	Heap	Stack	Stack VM	Total VM
Process 1 (25994): aprun	2.33MB	167.91KB	2.25MB	18.30KB	92.00KB	14.87MB
@syscall_library@-64	1476	264				
aprun	434.52KB	72.15KB				
ld-linux-x86-64.so.2	117.62KB	4.19KB				
libc.so.6	1339.36KB	34.88KB				
libdl.so.2	7.17KB	920				
libgcc_s.so.1	81.51KB	1544				
libnss_files.so.2	40.71KB	1872				
libpthread.so.0	86.49KB	17.88KB				
libtuhook-64.so	277.89KB	34.32KB				