SGI Developer Tools Update

Jim Galarowicz

Manager of Performance Tools and Debugger Tools Group

SGI Developer Tools Update

- Topics covered in this presentation
  - – Introduction to WorkShop and SpeedShop
  - – What are the current releases?
  - – Feature highlights of the current releases.
  - – Features scheduled in the next releases.
  - – Appendix with screen shots and examples

SGI Developer Tools Update

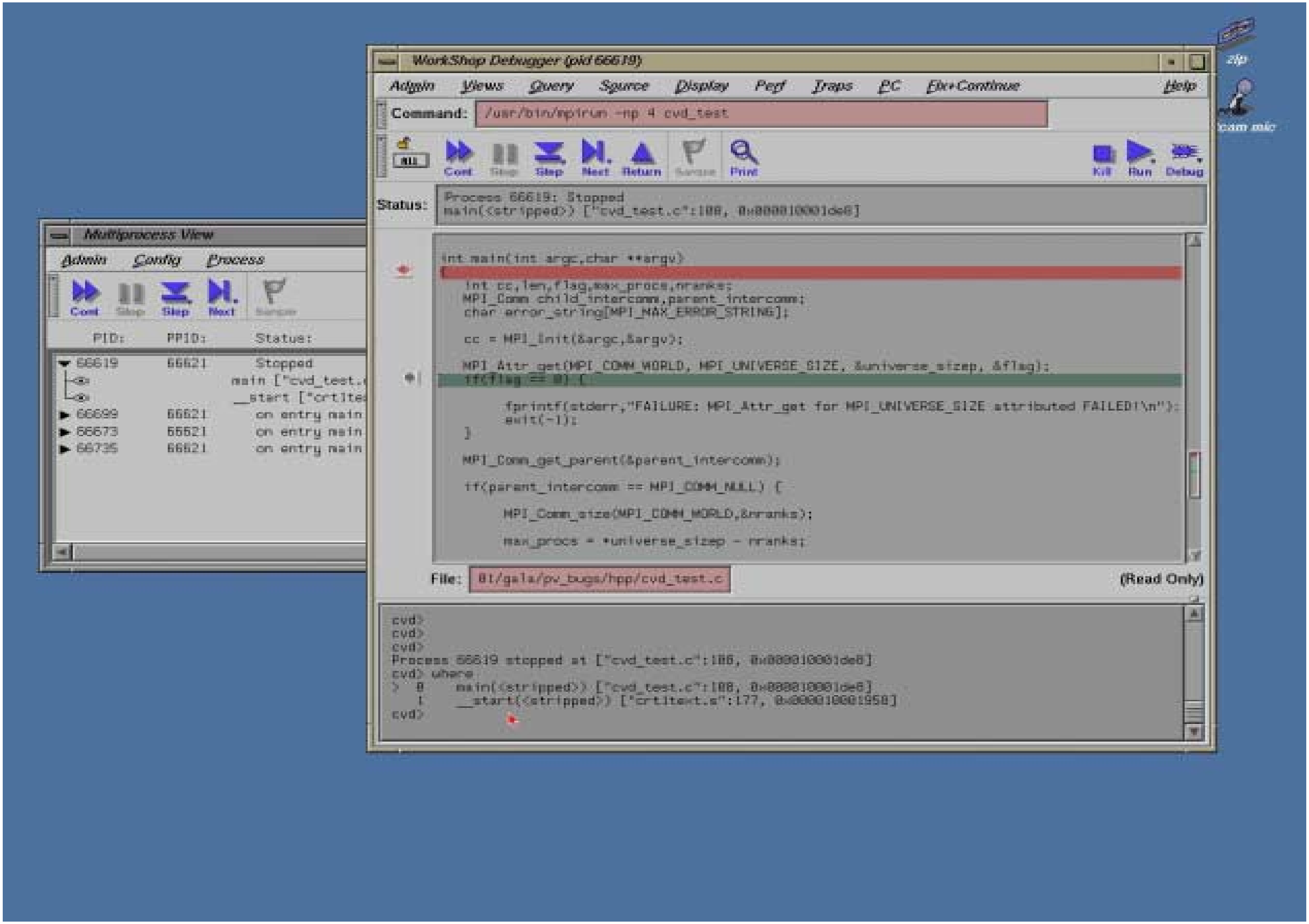
- Introduction to WorkShop and cvd
  - WorkShop is collection of tools
    - cvd - GUI based source level debugger
    - dbx - command line source level debugger
    - cvcov - coverage tool - what parts of my program are being executed
    - cvperf - performance analysis viewer
  - cvd <executable> or dbx <executable>
  - cvd mpirun -args -np 64 <mpi_executable>

SGI Developer Tools Update

Introduction to WorkShop and cvd

– cvd features and capability

- OpenMP, MPI for single system image, pthreads, shmem, MP mixed codes
- Fortran (f90,f77), C++, C, Ada, mixed lang. codes
- o32, n32, and 64 bit ABI’s
- specialized views for source, data, instruction, register information
- parallel view to control all or specific processes and/or threads



Command: /usr/bin/mpirun -np 4 cvd_test

Status:
```
Process 66619: Stopped
main(<stripped>) ["cvd_test.c":108, 0x00001000de8]
```

```
int main(int argc,char **argv)
{
   int cc,len,flag,max_procs,nranks;
   MPI_Comm child_intercomm,parent_intercomm;
   char error_string[MPI_MAX_ERROR_STRING];

   cc = MPI_Init(&argc,&argv);

   MPI_Attr_get(MPI_COMM_WORLD, MPI_UNIVERSE_SIZE, &universe_sizep, &flag);
   if(flag == 0) {

        fprintf(stderr,"FAILURE: MPI_Attr_get for MPI_UNIVERSE_SIZE attributed FAILED!\n");
        exit(-1);
   }

   MPI_Comm_get_parent(&parent_intercomm);

   if(parent_intercomm == MPI_COMM_NULL) {

        MPI_Comm_size(MPI_COMM_WORLD,&nranks);

        max_procs = *universe_sizep - nranks;
```

File: 0l/gala/pv_bugs/hpp/cvd_test.c (Read Only)

```
cvd>
cvd>
cvd>
Process 66619 stopped at ["cvd_test.c":108, 0x00001000de8]
cvd> where
>  0    main(<stripped>) ["cvd_test.c":108, 0x00001000de8]
   1    __start(<stripped>) ["crtltext.s":177, 0x00001000958]
cvd>
```

```
    PID:       PPID:       Status:
▼ 66619       66621        Stopped
  ├                     main ["cvd_test.c
  └                      __start ["crtlte
▶ 66699       66621        on entry main
▶ 66673       66621        on entry main
▶ 66735       66621        on entry main
```

What are the current releases?

- Released Jan 8th, 2002
- WorkShop 2.9.1
  - Download WS 2.9.1 from supportfolio - current patch 4531
- SpeedShop 1.4.3
  - Download 1.4.3 from supportfolio - current patch 4532
- dbx 7.3.3
  - Download 7.3.3 from supportfolio - current patch 4530

- http://support.sgi.com/colls/patches/tools/relstream/index.html

Feature highlights in the new WorkShop/dbx release

- Visualizing arrays of pointers/derived types
- F90 debugging improvements
- Pthread debugging improvements
- Auto-dereferencing for pointers of simple types
- Improvements to the speed of execution of cvd
- Other Improvements (merge, other)
- Coverage tool improvements (cvcov)

Feature highlights in the new WS/dbx release

- *F90 debugging improvements*
  - Array browser allows display of derived types
  - All views can now display data declared outside a F90 internal procedure while in the internal procedure.
  - Indexing Fortran90 pointer-arrays within a derived type now works. For example, 'dt%array(i,j)’.
  - Other problems have been fixed in the release and in patch 4531 (WS) and 4530 (dbx).

Feature highlights in the new WS/dbx release

- *Pthread debugging improvement*
  - Stepping over pthread_create now has consistent behavior.   Prior to the fix you could get stuck in “Running state”
  - Fixes to Multiprocess view (MPView).

Feature highlights in the new WS/dbx release

- *Improvements - speed of execution of cvd*
  - Some key server routines were reworked to be faster
  - Raised the compiler optimization level in building cvd
  - Stepping over (next) is faster

Feature highlights in the new WS/dbx release

Other Improvements

- *Due to merging cvd/dbx server*
  - Several cases closed that were fixed in one or the other
  - 64 bit debugging improvements
  - Common source base for fixing server problems.
- *Gui Area*
  - Double clicking from Trap Manager positions source
  - Wide character WorldView fixes - can see comments
  - Clearcase support

Feature highlights in the new WS/dbx release

WorkShop coverage tool improvements (cvcov) - reports execution statistics.

- Improved C++ support (lang:std)
- Updated the default exclusion file
- Reduce cvcov memory leaks - purify,SS
- Improve cvcov speed of execution on large appl.

SGI Developer Tools
Update

WorkShop and dbx

Features planned for the next release.

Features planned for the next WS/dbx release.

- *GUI data access improvements*
  - Single mouse action process & data navigation
  - Reengineered Multiprocess Explorer (was MPView)
  - Reengineered Data Explorer (was Structure Browser)
- *OpenMP debugging improvements*
- *cvd/dbx memory usage improvements*

Features planned for the next WS/dbx release.

- *Process & data navigation across cvd*
  - One mouse action access to data
  - Right mouse button down give default action
  - Left or Right mouse button hold gives dynamic menu
  - Data display window for lightweight data display (reusable or throw away)
  - Reusable can access previous data panels

Features planned for the next WS/dbx release.

- *Reengineered Multiprocess Explorer*
  - Show MPI rank in Multiprocess Explorer
  - Optional viewing of information in Mulitiprocess Explorer
- *Reengineered Data Explorer*
  - Name and minimal type info list for selection
  - Data navigation capabilities
  - Hoping to add multiprocess/thread data navigation

Features planned for the next WS/dbx release.

- *Improved OpenMP support*
  - Show private data w/o frame changing to main program and other improvements
- *No significant memory growth in cvd/dbx on long sessions and re-runs*
  - New garbage collection algorithm to improve memory usage
  - String table re-engineering

SpeedShop

SGI Performance Analysis Tools

SGI Developer Tools Update

Introduction to SpeedShop and cvperf

- SpeedShop is collection of performance analysis tools
  - ssrun - SpeedShop experiment driver
  - cvperf - GUI performance analysis viewer
  - prof - text based performance analysis viewer
- ssrun -experiment <executable>
- mpirun -np 4 ssrun -mpi <mpi_executable>
- cvperf <experiment_file>

Current release Information

SpeedShop 1.4.3

- Released Jan 8th, 2002
- Is delivered as part of ProDev WorkShop 2.9.1
- Download 1.4.3 from supportfolio
- Current patch 4532
- http://support.sgi.com/colls/patches/tools/relstream/index.html

SpeedShop 1.4.3

SpeedShop 1.4.3 introduces two new performance experiments:

- MPI Experiment (“ssrun –mpi”)
- NUMA Experiment (“ssrun –numa”)

Includes numerous bug fixes

New MPI experiment

MPI experiment answers four basic questions:

- Which MPI function was called?
- Who made the call?
- Where was the call made? (to the source line)
- How long did the call take?

New NUMA experiment

NUMA experiment answers these questions:

- How often do I access memory on my own NUMA node?
- Where am I accessing memory from a remote NUMA node?
- Are my placement directives working?

SGI Developer Tools Update

SpeedShop

Features planned for the next release.

SGI Developer Tools Update

SpeedShop features for next release

- *R16k Support*
- *Improved by pthread data collection*
- *Pthread data by thread in cvperf*
- *Multiple starts and stops of data collection*
- *MPI and NUMA experiment improvements*
- *Bug fixes and other improvements*

Questions?

SGI Developer Tools Update

Jim Galarowicz

jeg@sgi.com

Developer Tools Appendix

SGI Developer Tools Update
Additional Slides
showing features discussed in the
Tools Presentation

SGI Developer Tools Update

WorkShop
Appendix Information

Feature highlights in the current WorkShop/dbx release

- WorkShop coverage tool improvements (cvcov) - reports execution statistics.
  - Improved C++ support (lang:std)
  - Updated the default exclusion file
  - Reduce cvcov memory leaks - purify,SS
  - Improve cvcov speed of execution on large appl.

Feature highlights in the current WorkShop/dbx release

- Coverage tool usage (cvcov)
  - cvcov runinstr a.out
  - cvcov -mktest -cmd “a.out -d”
  - cvcov runtest test0000
  - cvcov lssource funcname test0000 . List of annotated src
  - cvcov lssum test0000 . Summary of coverage
  - cvcov lscall test0000 . Lists function call graph

Feature highlights in the current WorkShop/dbx release

Coverage tool usage (cvcov)

cvcov runinstr hashTest

```
runinstr command: /usr/sbin//cvinstr -coverage
   /usr/WorkShop/usr/lib/WorkShop/Tester/default_instr_file  -addlibs
   libss.so:libssrt.so -directory
   /data/clink/a01/gala/pv_bugs/swift_probs/ver##0 "hashTest"
       instrumenting /lib32/rld
       instrumenting /usr/lib32/mips3/libssrt.so
       instrumenting /usr/lib32/mips3/libss.so
       instrumenting /usr/lib32/mips3/libc.so.1
       instrumenting hashTest
cvcov: Instrument "hashTest" of version "0" succeeded.
```

cvcov mktest -cmd hashTest

cvcov: Made test directory: "/data/clink/a01/gala/pv_bugs/swift_probs/test0000"

Feature highlights in the current WorkShop/dbx release

Coverage tool usage (cvcov)

cvcov runtest test0000

```
cvcov: Running test
   "/data/clink/a01/gala/pv_bugs/swift_probs/test0000" ...
/data/clink/a01/gala/pv_bugs/swift_probs//ver##0/hashTest_Instr
found: abc with value: 0
found: def with value: 1
found: alsdjk with value: 2
found: Smith with value: 3
found: june with value: 4
found: smith with value: 3
found: June with value: 4
Key: abc, value: 0
```

Feature highlights in the current WorkShop/dbx release

Coverage tool usage (cvcov)

```
cvcov lssum test0000
          Coverages          Covered     Total       % Coverage
             Weight
          ----------------------------------------------------------------------
          Function              8           39          20.51%        0.400
          Source Line       71         446         15.92%        0.200
          Branch             10          239          4.18%        0.200
          Arc                 20          176          11.36%        0.200
          Block              75          624         12.02%        0.000
          Weighted Sum                              14.50%        1.000
```

Feature highlights in the current WorkShop/dbx release

Coverage tool usage (cvcov)

cvcov lsfun test0000

```
Functions       Files   Counts
------------------------------------
is_ival isIval.c        0
is_rval isRval.c        0
make_hashtable  serv.c  1
free_hashtable  serv.c  1
insert_in_hash  serv.c  7
find_in_hash    serv.c  7
for_all_in_hash serv.c  1
hashUC  serv.c  14
```

Feature highlights in the current WorkShop/dbx release

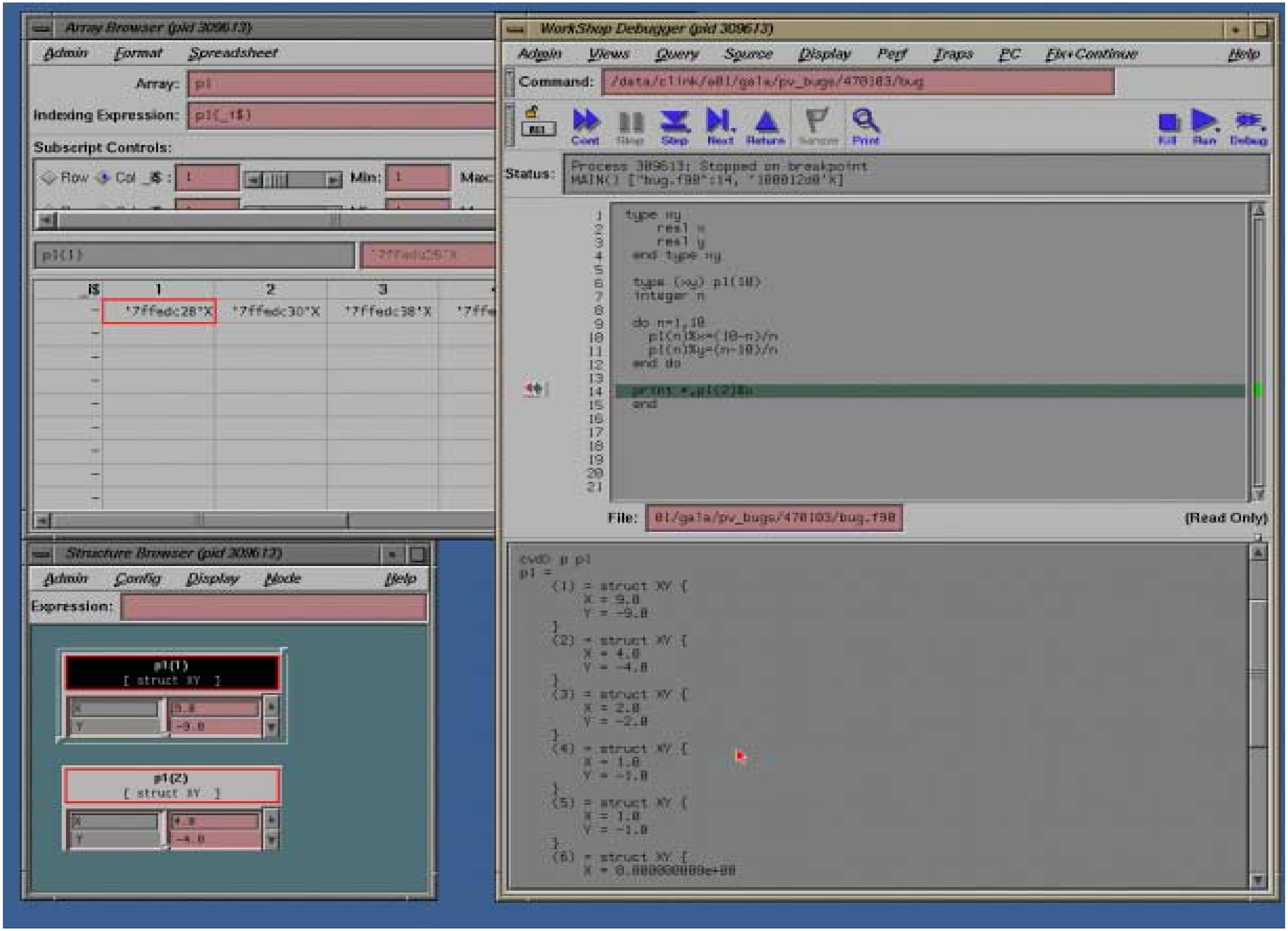
Coverage tool usage (cvcov)

cvcov lssource for_all_in_hash test0000

```
Counts  Source
--------------------------------------------------------------
        void for_all_in_hash(Hash* ht, void (*func)(const char *, void
    *))
1       {
         int   ja;
         Link* link;
257        for (ja = 0; ja < ht->num_buckets; ja++)
257          for (link = ht->buckets[ja].link; link; link=link->next )
5            (*func)(link->key, link->adt);
1       }
```

Feature highlights in the current WorkShop/dbx release

- *Visualizing arrays of pointers/derived types*
  - Bring up Array Visualizer and enter array name.
  - Double click on the entry you are interested in.
  - Structure or derived type is displayed in the structure browser.
  - Example on next slide.



Array: p1

Indexing Expression: p1(_i$)

Subscript Controls:

◇ Row ◆ Col _i$: 1 Min: 1 Max:

p1(1) '7ffedc28'X

```
 _i$         1              2              3         
  _   '7ffedc28'X    '7ffedc30'X    '7ffedc38'X    '7ffe
```

Command: /data/clink/a01/gala/pv_bugs/470103/bug

Status:

```
Process 309613: Stopped on breakpoint
MAIN() ["bug.f90":14, '100012d0'X]
```

```
type xy
    real x
    real y
 end type xy

 type (xy) p1(10)
 integer n

 do n=1,10
   p1(n)%x=(10-n)/n
   p1(n)%y=(n-10)/n
 end do

 print *,p1(2)%x
 end
```

File: 01/gala/pv_bugs/470103/bug.f90 (Read Only)

```
cvd> p p1
p1 =
     (1) = struct XY {
         X = 9.0
         Y = -9.0
     }
     (2) = struct XY {
         X = 4.0
         Y = -4.0
     }
     (3) = struct XY {
         X = 2.0
         Y = -2.0
     }
     (4) = struct XY {
         X = 1.0
         Y = -1.0
     }
     (5) = struct XY {
         X = 1.0
         Y = -1.0
     }
     (6) = struct XY {
         X = 0.000000000e+00
```

Expression:

p1(1)
[struct XY]

X 9.0
Y -9.0

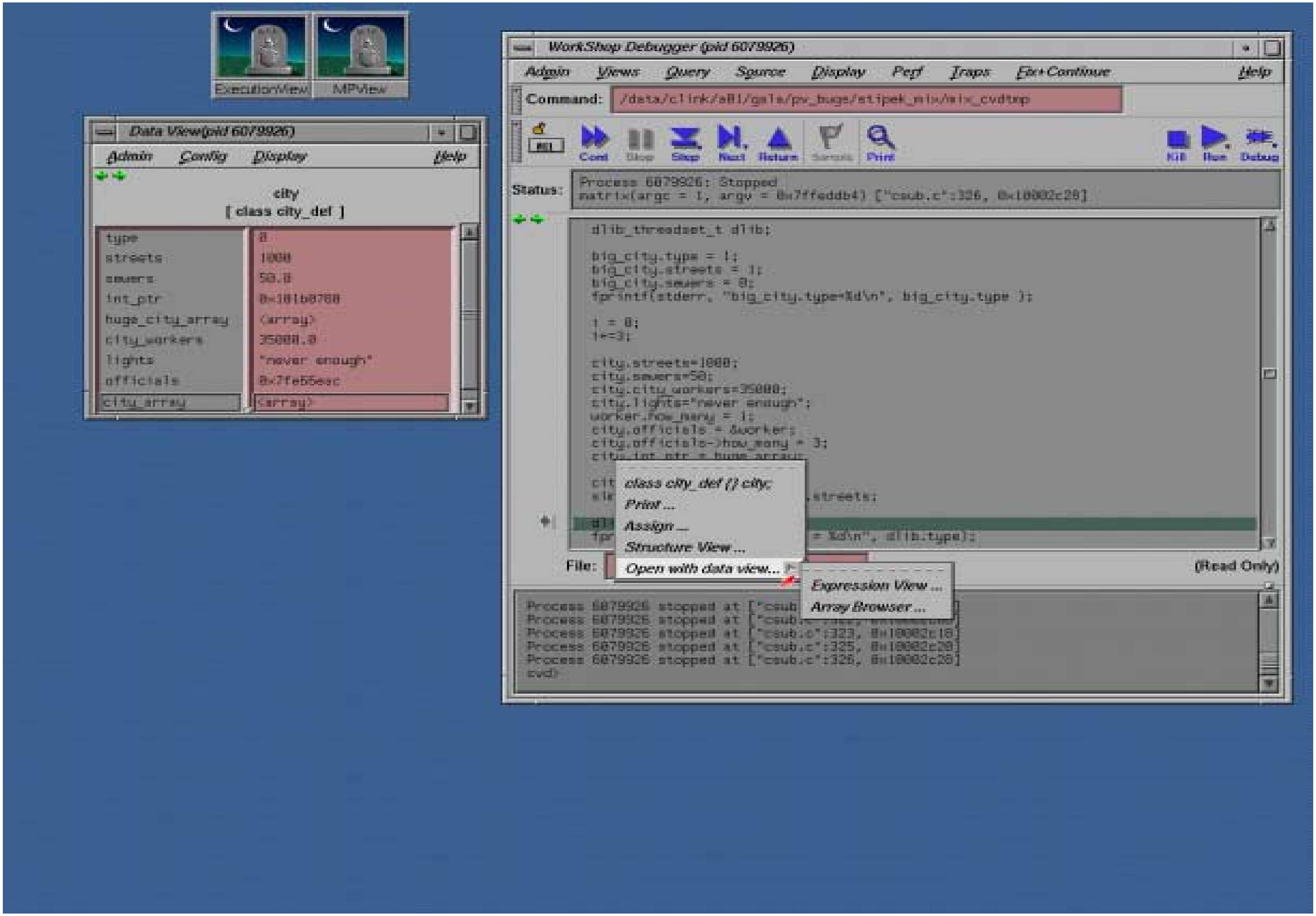
p1(2)
[struct XY]

X 4.0
Y -4.0

Features planned for the next WS/dbx release.

Data navigation across cvd views

- One mouse action access to data
- Right mouse button down give default action
- Left or Right mouse button hold gives dynamic menu
- Data display window for lightweight data display (reusable or throw away)
- Reusable can access previous data panels
  - Example on next slide



city
[class city_def]

type	0
streets	1000
sewers	50.0
int_ptr	0x101b0700
huge_city_array	(array)
city_workers	35000.0
lights	"never enough"
officials	0x7fe55esc
city_array	(array)

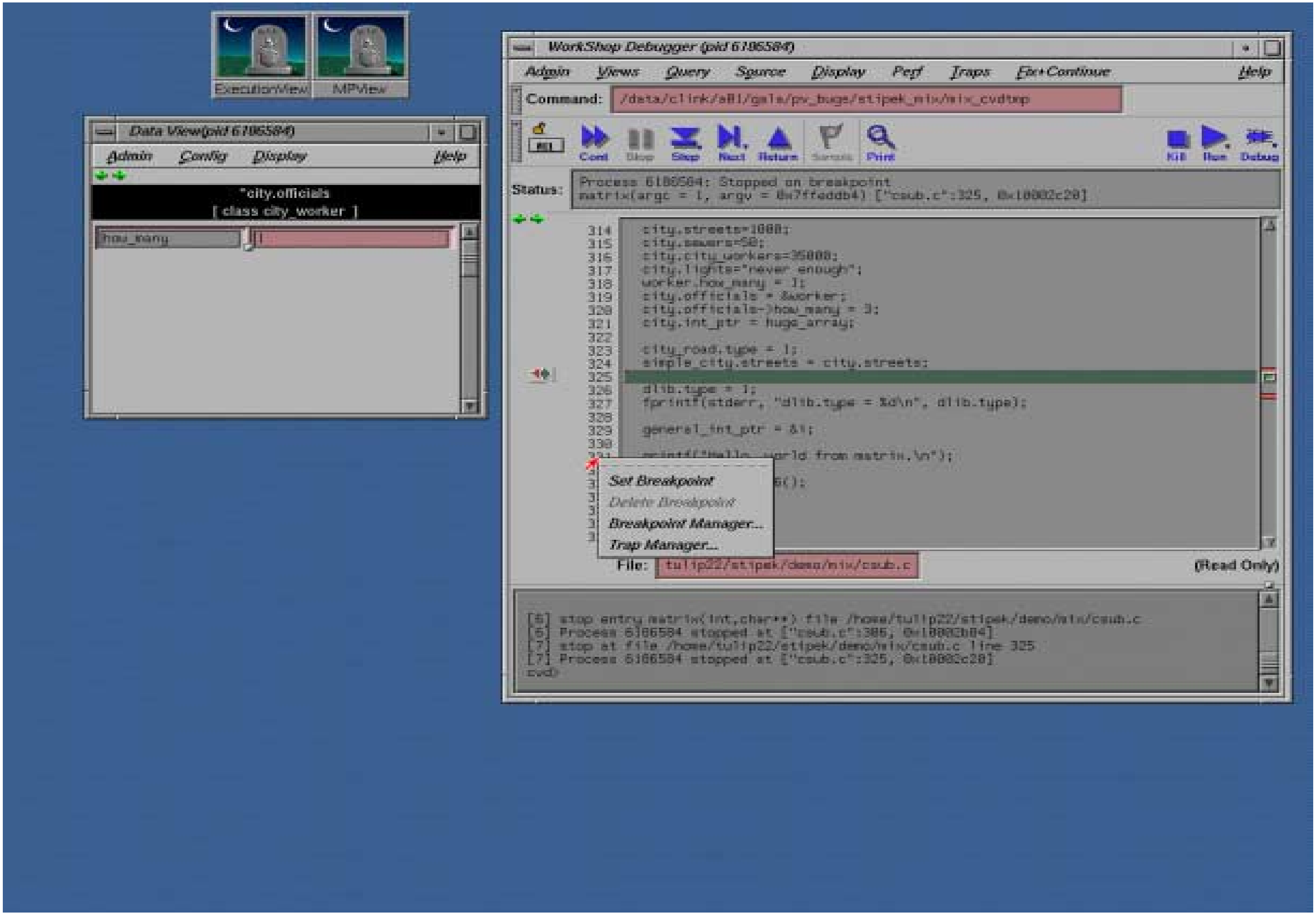
Command: /data/clink/a81/gala/pv_bugs/stipek_mix/mix_cvdtmp

Status:
```
Process 6079926: Stopped
matrix(argc = 1, argv = 0x7ffeddb4) ["csub.c":326, 0x10002c28]
```

```
dlib_threadset_t dlib;

big_city.type = 1;
big_city.streets = 1;
big_city.sewers = 0;
fprintf(stderr, "big_city.type=%d\n", big_city.type );

i = 0;
i+=3;

city.streets=1000;
city.sewers=50;
city.city_workers=35000;
city.lights="never enough";
worker.how_many = 1;
city.officials = &worker;
city.officials->how_many = 3;
city.int_ptr = huge_array;

cit
els                               .streets;

dl
fpr                               = %d\n", dlib.type);
```

File:

```
Process 6079926 stopped at ["csub
Process 6079926 stopped at ["csub
Process 6079926 stopped at ["csub.c":323, 0x10002c18]
Process 6079926 stopped at ["csub.c":325, 0x10002c20]
Process 6079926 stopped at ["csub.c":326, 0x10002c28]
cvd>
```



*city.officials
[class city_worker]

how_many: 1

Command: /data/clink/a81/gals/pv_bugs/stipek_mix/mix_cvdtmp

Status:
```
Process 6186584: Stopped on breakpoint
matrix(argc = 1, argv = 0x7ffeddb4) ["csub.c":325, 0x10002c20]
```

```
city.streets=1000;
city.sewers=50;
city.city_workers=35000;
city.lights="never enough";
worker.how_many = 1;
city.officials = &worker;
city.officials->how_many = 3;
city.int_ptr = huge_array;

city_road.type = 1;
simple_city.streets = city.streets;

dlib.type = 1;
fprintf(stderr, "dlib.type = %d\n", dlib.type);

general_int_ptr = &1;

printf("Hello, world from matrix.\n");

6();
```

File: tulip22/stipek/demo/mix/csub.c

```
[6] stop entry matrix(int,char**) file /home/tulip22/stipek/demo/mix/csub.c
[6] Process 6186584 stopped at ["csub.c":306, 0x10002b84]
[7] stop at file /home/tulip22/stipek/demo/mix/csub.c line 325
[7] Process 6186584 stopped at ["csub.c":325, 0x10002c20]
cvd>
```

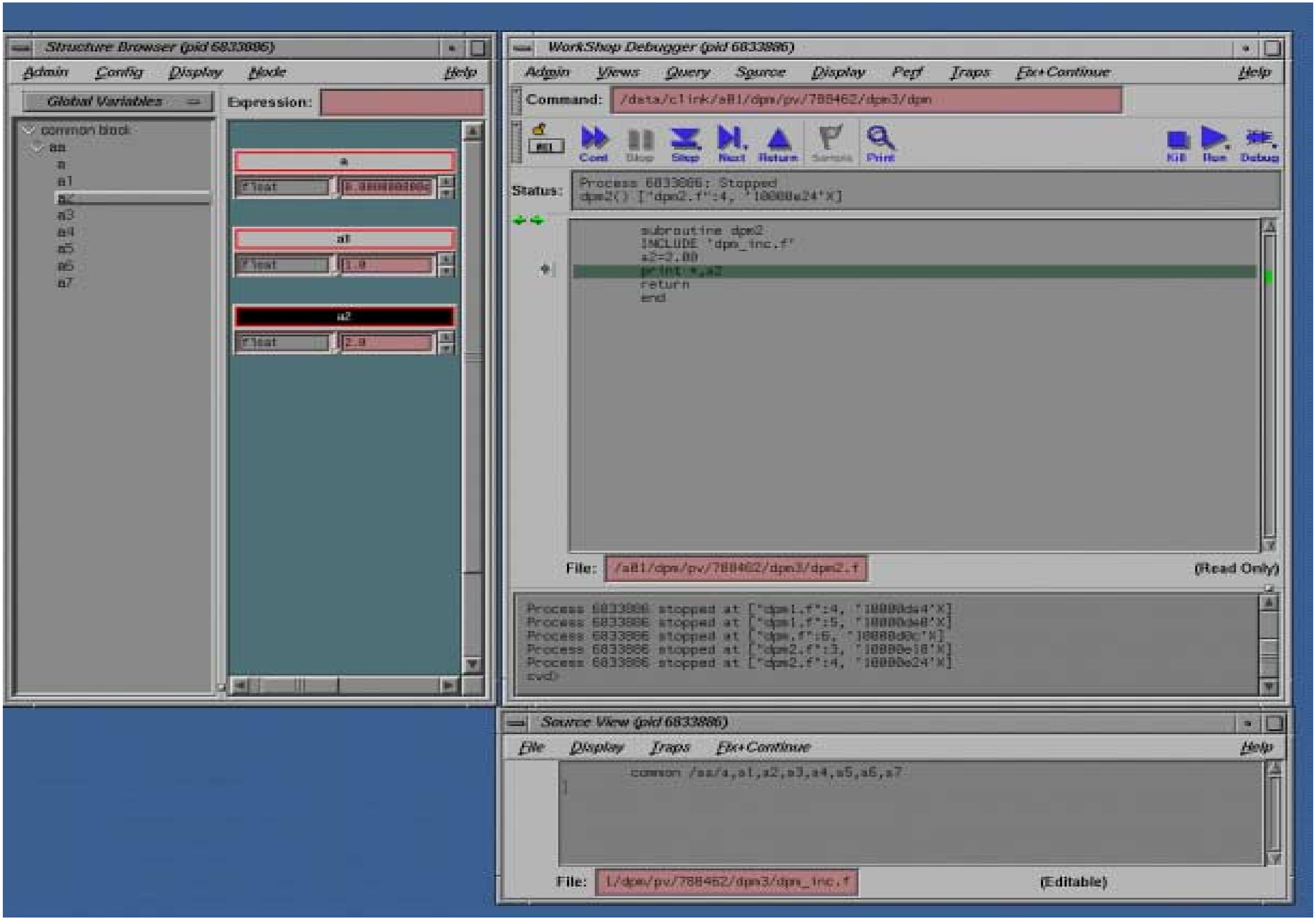
Features planned for the next WS/dbx release.

Reengineered Multiprocess Explorer

- Show MPI rank in Multiprocess Explorer (was MPView)
- More readable Multiprocess Explorer

Reengineered Data Explorer View

- Name and minimal type information list for selection
- Data navigation capabilities
- Hoping to add multiprocess/thread data navigation
  - Example on next slide



Command: /data/clink/a81/dpm/pv/788462/dpm3/dpm

```
Process 6833886: Stopped
dpm2() ["dpm2.f":4, '10000e24'X]
```

```
      subroutine dpm2
      INCLUDE 'dpm_inc.f'
      a2=2.00
      print *,a2
      return
      end
```

File: /a81/dpm/pv/788462/dpm3/dpm2.f

```
Process 6833886 stopped at ["dpm1.f":4, '10000de4'X]
Process 6833886 stopped at ["dpm1.f":5, '10000de8'X]
Process 6833886 stopped at ["dpm.f":6, '10000d0c'X]
Process 6833886 stopped at ["dpm2.f":3, '10000e18'X]
Process 6833886 stopped at ["dpm2.f":4, '10000e24'X]
cvd>
```

```
      common /aa/a,a1,a2,a3,a4,a5,a6,a7
```

File: l/dpm/pv/788462/dpm3/dpm_inc.f

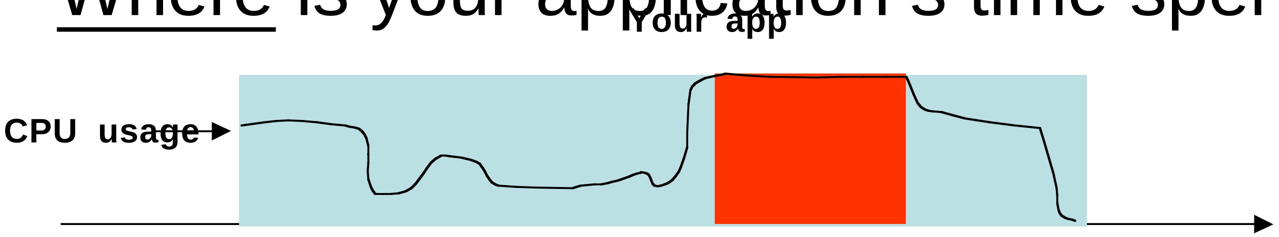
SGI Developer Tools Update

SpeedShop

Appendix Information

What Is SpeedShop?

- *A collection of tools to determine:*
  - Where is your application's time spent?



  - How is your application's time spent?

  - What are your application's bottlenecks?

Why Use SpeedShop?

- *It will help you:*
  - Minimize application development time
  - Eliminate bottlenecks and bugs
  - Maximize your application’s overall performance

- *How invoke SpeedShop:*
  - ssrun -<expr-type> <executable>

What SpeedShop will tell you?

- **SpeedShop reports statistical data**
  - **Function name, source file name, line number, and a statistical data that depends on the experiment you're running**
  - **It includes system functions**
- **Cvperf will let you browse, via a GUI, into your code**
- **Prof will gives a text based quick report**

SpeedShop 1.4.3

SpeedShop 1.4.3 introduces two new performance experiments:

- MPI Experiment (“ssrun –mpi”)
- NUMA Experiment (“ssrun –numa”)

Includes numerous bug fixes

New MPI experiment

MPI experiment answers four basic questions:

- Which MPI function was called?
- Who made the call?
- Where was the call made? (to the source line)
- How long did the call take?

MPI Experiment Example

How do I use it?

- – NAS CG Parallel Benchmark (MPI)
- – Origin 2000 with 16 300Mhz R12000
- – mpirun -np 4 ssrun -mpi cg.A.4

```
% mpirun -np 4 ssrun -mpi cg.A.4
...
% prof cg.A.4.mpi.f1384250
-------------------------------------------------------------------------
SpeedShop profile listing generated Wed Jan 23 13:33:52 2002
...
-------------------------------------------------------------------------
Summary of MPI tracing data (mpi)--
                      5044: Total Traced MPI calls
-------------------------------------------------------------------------
Callee list, in descending order by time taken in MPI call
-------------------------------------------------------------------------
   Seconds      Calls  MPI Function

     4.602      1680  MPI_Wait
     1.046         1  MPI_Finalize
     0.374      1680  MPI_Send
     0.204      1680  MPI_Irecv
     0.061         1  MPI_Init
     0.000         1  MPI_Reduce
     0.000         1  MPI_Barrier
```

MPI Example (Continued)

```
---------------------------------------------------------------------------
Call site list, in descending order by time taken in MPI call
---------------------------------------------------------------------------
  Seconds     Calls  MPI Function            Function (dso: file, line)

    2.933       400  MPI_Wait               conj_grad (cg.A.4: cg.f, 1177)
    1.409       400  MPI_Wait               conj_grad (cg.A.4: cg.f, 1150)
    1.046         1  MPI_Finalize           cg (cg.A.4: cg.f, 571)
    0.180       400  MPI_Send               conj_grad (cg.A.4: cg.f, 1170)
    0.169       400  MPI_Send               conj_grad (cg.A.4: cg.f, 1147)
    0.133       400  MPI_Irecv              conj_grad (cg.A.4: cg.f, 1161)
    0.099        16  MPI_Wait               conj_grad (cg.A.4: cg.f, 1059)
    0.069        16  MPI_Wait               conj_grad (cg.A.4: cg.f, 1361)
    0.061         1  MPI_Init               initialize_mpi (cg.A.4: cg.f, 594)
    0.048       400  MPI_Irecv              conj_grad (cg.A.4: cg.f, 1139)
    0.041        16  MPI_Wait               conj_grad (cg.A.4: cg.f, 1333)
    0.037       400  MPI_Wait               conj_grad (cg.A.4: cg.f, 1275)
    0.016       400  MPI_Wait               conj_grad (cg.A.4: cg.f, 1221)
    0.013       400  MPI_Irecv              conj_grad (cg.A.4: cg.f, 1209)
    0.008        16  MPI_Send               conj_grad (cg.A.4: cg.f, 1354)
    0.006        16  MPI_Send               conj_grad (cg.A.4: cg.f, 1330)
    0.006       400  MPI_Irecv              conj_grad (cg.A.4: cg.f, 1264)
    0.005       400  MPI_Send               conj_grad (cg.A.4: cg.f, 1217)
    0.005       400  MPI_Send               conj_grad (cg.A.4: cg.f, 1272)
    0.002        16  MPI_Irecv              conj_grad (cg.A.4: cg.f, 1345)
    0.001        16  MPI_Irecv              conj_grad (cg.A.4: cg.f, 1322)
    0.000        16  MPI_Irecv              conj_grad (cg.A.4: cg.f, 1386)
    0.000         1  MPI_Reduce             cg (cg.A.4: cg.f, 509)
    0.000        16  MPI_Irecv              conj_grad (cg.A.4: cg.f, 1048)
    0.000        15  MPI_Wait               cg (cg.A.4: cg.f, 474)
    0.000        16  MPI_Send               conj_grad (cg.A.4: cg.f, 1056)
    0.000        16  MPI_Send               conj_grad (cg.A.4: cg.f, 1394)
    0.000        15  MPI_Irecv              cg (cg.A.4: cg.f, 463)
    0.000        15  MPI_Send               cg (cg.A.4: cg.f, 471)
    0.000         1  MPI_Barrier            cg (cg.A.4: cg.f, 411)
    0.000        16  MPI_Wait               conj_grad (cg.A.4: cg.f, 1397)
    0.000         1  MPI_Wait               cg (cg.A.4: cg.f, 376)
    0.000         1  MPI_Irecv              cg (cg.A.4: cg.f, 365)
    0.000         1  MPI_Send               cg (cg.A.4: cg.f, 373)
```

New NUMA experiment

NUMA experiment answers these questions:

– How often do I access memory on my own NUMA node?

– Where am I accessing memory from a remote NUMA node?

– Are my placement directives working?

NUMA experiment:
How does it work?

NUMA experiment does these items:

- *Statistically samples memory accesses*
- *Looks for a “computable” memory access*
- *Stores “numa info” for each sample*
  - see SpeedShop portion of Appendix for additional information

NUMA Example

- *How do I use it?*
  - – NAS CG Parallel Benchmark (OpenMP)
  - – Origin 2000 with 16 300Mhz R12000

```
% setenv OMP_NUM_THREADS 4
% ssrun -numa cg.A
...
% prof -source cg.A.numa.p1350994
-------------------------------------------------------------------------
SpeedShop profile listing generated Wed Jan 23 13:18:40 2002
...
-------------------------------------------------------------------------
Summary of NUMA memory profiling data (numa)--
 Secondary cache D misses (26): Counter Name (Number)
                        100: Counter Average Overflow
                     101066: Sampled Memory Accesses
                      76615: Remote Memory Accesses
                     75.807: Percent Remote Memory Accesses
                      1.436: Average ccNUMA Routing Distance
-------------------------------------------------------------------------
Function list, in descending order by percent remote memory accesses
-------------------------------------------------------------------------
   Sampled     Remote    Pct Rmt    Avg Dist  Function (dso: file, line)

...
    100739      76465     75.904       1.437  conj_grad (cg.A: cg.c, 374)
...
```

NUMA Example

```
------------------------------------------------------------------------
Disassembly listing, annotated with NUMA memory profiling data
------------------------------------------------------------------------
...
conj_grad (cg.A: cg.c, 374):
...
429: /* rolled version */
430: #pragma omp for private(sum,k)
431:          for (j = 1; j <= lastrow-firstrow+1; j++) {
432:       sum = 0.0;
433:             for (k = rowstr[j]; k < rowstr[j+1]; k++) {
434:                         sum = sum + a[k]*p[colidx[k]];
435:             }
436:       w[j] = sum;
437:          }
438:
...
[ 434] 0x100057fc        0x8c440000   lw        a0,0(v0)
   ^------ 220 Sampled, 75.909% Remote, Avg Dist = 1.459 ------^
[ 434] 0x10005800        0x000420c0   sll       a0,a0,3
[ 434] 0x10005804        0xd42d0000   ldc1      $f13,0(at)
   ^------ 85 Sampled, 68.235% Remote, Avg Dist = 1.306 ------^
[ 434] 0x10005808        0x02042021   addu      a0,s0,a0
[ 434] 0x1000580c        0xd48e0000   ldc1      $f14,0(a0)
   ^------ 106 Sampled, 83.019% Remote, Avg Dist = 1.500 ------^
[ 434] 0x10005810        0x24210008   addiu     at,at,8
[ 434] 0x10005814        0x24420004   addiu     v0,v0,4
[ 434] 0x10005818        0x4cee69e1   madd.d    $f7,$f7,$f13,$f14
[ 434] 0x1000581c        0x12600046   beq       s3,zero,0x10005938
[ 434] 0x10005820        0000000000   nop
[ 434] 0x10005824        0x16800698   bne       s4,zero,0x10007288
[ 434] 0x10005828        0000000000   nop
[ 434] 0x1000582c        0x8c440000   lw        a0,0(v0)
   ^------ 351 Sampled, 74.074% Remote, Avg Dist = 1.416 ------^
[ 434] 0x10005830        0xd4220008   ldc1      $f2,8(at)
   ^------ 582 Sampled, 77.491% Remote, Avg Dist = 1.474 ------^
[ 434] 0x10005834        0xd4230000   ldc1      $f3,0(at)
[ 434] 0x10005838        0x000420c0   sll       a0,a0,3
[ 434] 0x1000583c        0x8c430004   lw        v1,4(v0)
   ^------ 57 Sampled, 56.140% Remote, Avg Dist = 1.105 ------^
...
```

NUMA Experiment Notes & Caveats

- – Only supported on Origin 2000/3000
- – No GUI available (CVPERF not supported), only prof
- – Storage requirements can be excessive (32 bytes per sample)
- – Do other optimizations first!
  - First Order: algorithm selection
  - Second Order: algorithm implementation details
  - Third Order: NUMA placement