Program Development Tools

Ruud van der Pas

SiliconGraphics

European High Performance Computing Team

Topics

- ***Source code management***
- ***Make***
- ***System Monitoring Tools***
- ***The SpeedShop profiler***
- ***Performance Boosters***

Using these tools will save you time and frustration

Source Code Management

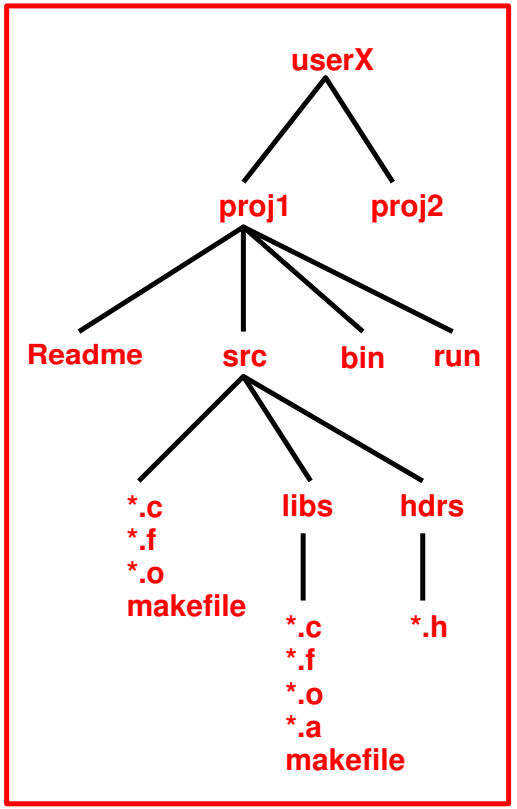
RCS/SCCS

Source Files

* Tree Organization

* Small Modules
  * fsplit / split
  * Faster Compile
  * Less Resources To Compile
  * Code Study
  * Optimization Flexibility

* Documentation
  * RCS/SCCS
  * Make
    * Readme



RCS (Revision Control System)

- *Automatic Documentation*
- *Roll–back*
- *100% "Oops" free*
- *Easy to use*

```
ruud@emmen 86% mkdir RCS
ruud@emmen 87% ls -lt main.f
-rw-r--r--    1 ruud     user    72 Mar 21 19:02 main.f
ruud@emmen 88% ci main.f
RCS/main.f,v  <--  main.f
enter description, terminated with single '.' or end of
file:
NOTE: This is NOT the log message!
>> Key new application
>> .
initial revision: 1.1
done
ruud@emmen 89% co main.f
RCS/main.f,v  -->  main.f
revision 1.1
done
ruud@emmen 90% rcsdiff main.f
===================================================================
RCS file: RCS/main.f,v
retrieving revision 1.1
diff -r1.1 main.f
ruud@emmen 91%
```

RCS Commands

- creation: (mkdir RCS) ci <file>
- versions: rlog <file>
- add new: ci <file>
- retrieve: co –r<version> <file>
- edit: co –l<version> <file>
- diff: rcsdiff –r<version> –r<version> <file>

SCCS

- Automatic Documentation
- Roll Back
- 100% "oops free"
- admin / get / delta / prs
- Make Integrated

```
admin -ifile1.c  s.file1.c
make -g t
```

SCCS Commands

- creation: admin –i<file> s.<file>
- versions: prs s.<file>
- add new: delta s.<file>
- retrieve: get –r<version> s.<file>
- edit: get –e –r<version> s.<file>
- diff: sccsdiff –r<version> –r<version> s.<file>

Make

Make

- Automatic Documentation
- Minimizes compiles + compiles in parallel (!)
- 100% "oops free"

```
pgm:main.o sub1.o sub2.o
<tab>   f77 -o pgm main.o sub1.o sub2.o \
        -64 -lm

sub1.o: sub1.f:
<tab>   f77 -c -O3 -64 -mips4 sub1.f

sub2.o: sub2.f
<tab>   f77 -c -O3 -64 -mips4 -static sub2.f

clean:
<tab>  -rm -f pgm main.o sub1.o sub2.o
```

Make – Example

```
# This make file uses symbols

EXE      = pgm
OBJECTS = main.o sub1.o sub2.o
FFLAGS   = -O3 -mips4 -64
LDFLAGS = -lm -64

default: $(EXE)

$(EXE): $(OBJECTS)
<tab>    f77 -o $(EXE) $(OBJECTS) $(LDFLAGS)

.f.o:
<tab>    f77 -c $(FFLAGS) $*.f

sub2.o: sub2.f
<tab>    f77 -c $(FFLAGS) -static $*.f

clean:
<tab>   @rm -f $(OBJECTS) $(EXE) core
```

Make – A More Elaborate Example

```
#
FC       = $(TOOLROOT)/usr/bin/f77
CC       = $(TOOLROOT)/usr/bin/cc
LD       = $(FC)
F77      = $(FC)
SWP      = swplist
RM       = /bin/rm -f
MP       =
ABI      = -n32
ISA      = -mips4
PROC     = ip27
ARCH     = $(MP) $(ABI) $(ISA)
OLEVEL   = -Ofast=$(PROC)
FOPTS    = -OPT:IEEE_arithmetic=3
COPTS    = -OPT:IEEE_arithmetic=3
FFLAGS   = $(ARCH) $(OLEVEL) $(FOPTS)
CFLAGS   = $(ARCH) $(OLEVEL) $(COPTS)
LIBS     = -lfastm
LDFLAGS  = $(ARCH) $(OLEVEL)
PROF     =

FOBJS    = f1.o f2.o f3.o
COBJS    = c1.o c2.o c3.o
OBJS     = $(FOBJS) $(COBJS)
EXEC     = executable

$(EXEC):         $(OBJS)
        $(LD) -o $@ $(PROF) $(LDFLAGS) $(QV_LOPT) $(OBJS) $(LIBS)

clean:
        $(RM) $(EXEC) $(OBJS)
        $(SWP) -c $(CFLAGS) $(QV_OPT) $(DEFINES) $<
```

```
.SUFFIXES:
.SUFFIXES: .o .F .c .f .swp

.F.o:
        $(FC)  -c $(FFLAGS) $(QV_OPT) $(DEFINES) $<

.f.o:
        $(FC)  -c $(FFLAGS) $(QV_OPT) $(DEFINES) $<

.c.o:
        $(CC)  -c $(CFLAGS) $(QV_OPT) $(DEFINES) $<

.F.swp:
        $(SWP) -c $(FFLAGS) $(QV_OPT) $(DEFINES) -WK,-cmp=$*.m $<

.f.swp:
        $(SWP) -c $(FFLAGS) $(QV_OPT) $(DEFINES) -WK,-cmp=$*.m $<

.c.swp:
        $(SWP) -c $(CFLAGS) $(QV_OPT) $(DEFINES) $<
```

Make has been parallelised

Speed–up your compiles:

make old standard, serial compilation
smake old standard, parallel compiles
pmake many extensions on old standard, parallel compiles

```
% pmake -JN  (make with "N" compile streams)
```

System Monitoring Tools

System Monitoring Tools

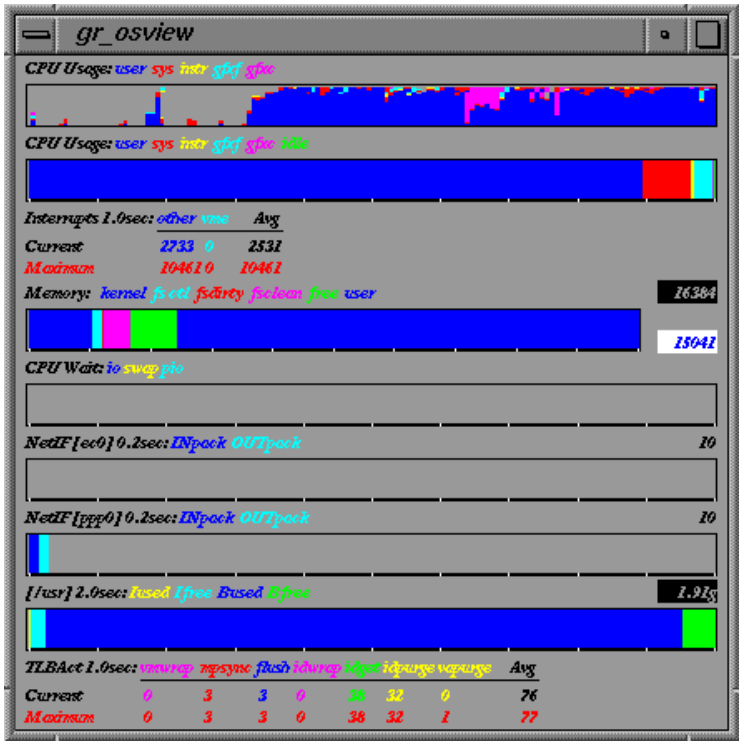
- *gr_osview (and osview)*
- *sar*
- *timex*

gr_osview

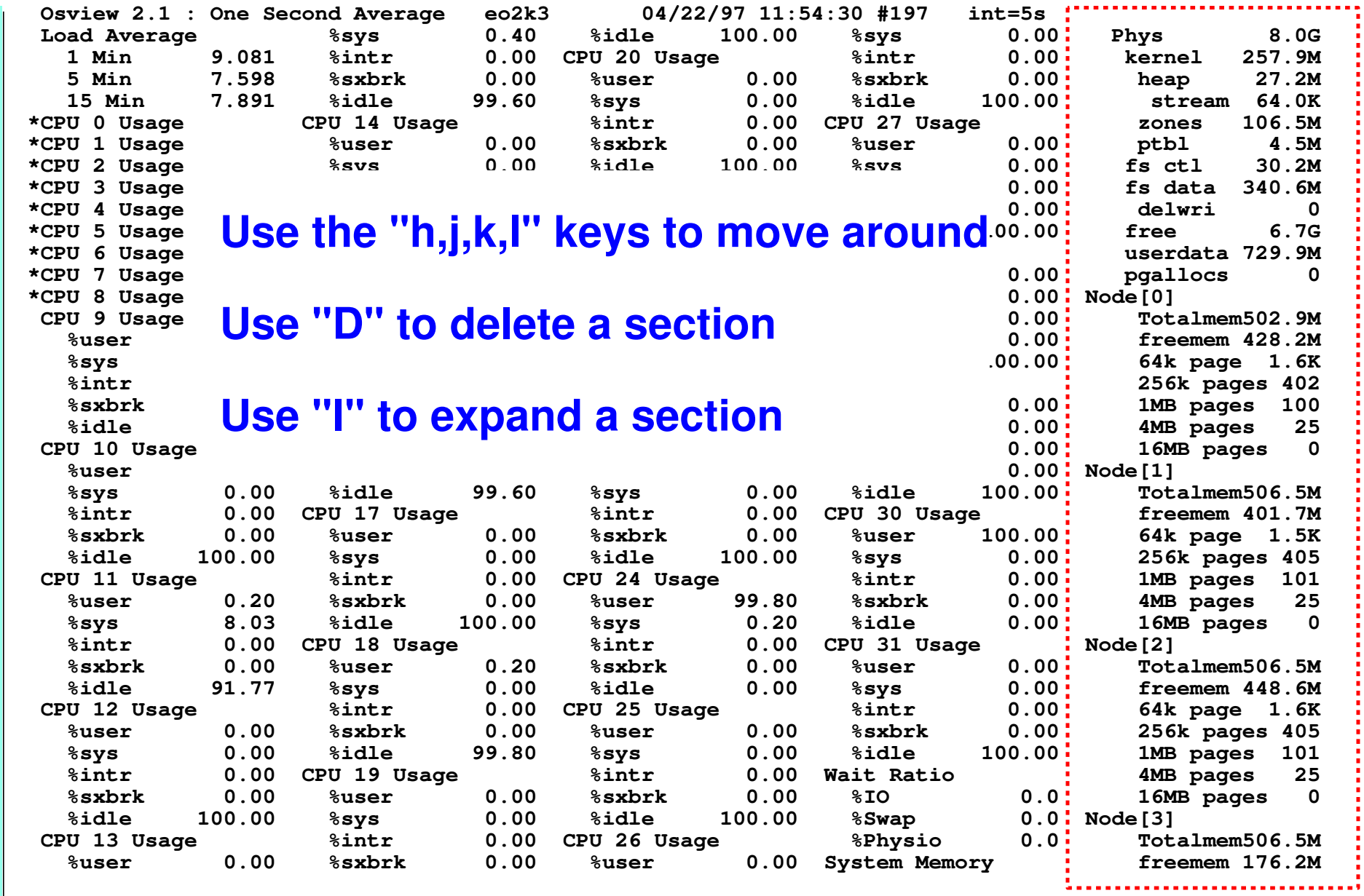
```
# Example configuration file
cpu strip
cpu
intr numeric max tracksum(10)
rmem tracksum
wait
netif
disk(/usr)
tlb numeric max tracksum(1)
```

Keywords:

```
strip        - display as a strip chart
numeric      - display numeric values
max          - add a maximum value numeric meter
max(n)       - add a maximum value meter with reset interval n
tracksum     - add an average value numeric meter
tracksum(n)  - add an average value meter with interval n
noborder     - suppress bar border
interval(n)  - set sample interval in base units
samples(n)   - set number of samples in strip chart

attack(f)       - set attack speed of bar
colors(c,...)   - set bar colors to use
maxcolor(b,f)   - set maximum value display colors
limcolor(b,f)   - set scale limit value display colors
sumcolor(b,f)   - set tracking sum display colors
lockscale(x)    - lock the scale value
creepscale      - autoscaling will only increase the scale
ticks(i,n)      - set strip chart tick marks
```



osview


Osview 2.1 : One Second Average eo2k3 04/22/97 11:54:30 #197 int=5s
Load Average
1 Min 9.081
5 Min 7.598
15 Min 7.891
*CPU 0 Usage
*CPU 1 Usage
*CPU 2 Usage
*CPU 3 Usage
*CPU 4 Usage
*CPU 5 Usage
*CPU 6 Usage
*CPU 7 Usage
*CPU 8 Usage
CPU 9 Usage
%user
%sys
%intr
%sxbrk
%idle
CPU 10 Usage
%user
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 11 Usage
%user 0.20
%sys 8.03
%intr 0.00
%sxbrk 0.00
%idle 91.77
CPU 12 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 13 Usage
%user 0.00
%sys 0.40
%intr 0.00
%sxbrk 0.00
%idle 99.60
CPU 14 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 17 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 99.60
CPU 18 Usage
%user 0.20
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 99.80
CPU 19 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 20 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 24 Usage
%user 99.80
%sys 0.20
%intr 0.00
%sxbrk 0.00
%idle 0.00
CPU 25 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 26 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 27 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 30 Usage
%user 100.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
CPU 31 Usage
%user 0.00
%sys 0.00
%intr 0.00
%sxbrk 0.00
%idle 100.00
Wait Ratio
%IO 0.0
%Swap 0.0
%Physio 0.0
System Memory
Phys 8.0G
kernel 257.9M
heap 27.2M
stream 64.0K
zones 106.5M
ptbl 4.5M
fs ctl 30.2M
fs data 340.6M
delwri 0
free 6.7G
userdata 729.9M
pgallocs 0
Node[0]
Totalmem 502.9M
freemem 428.2M
64k page 1.6K
256k pages 402
1MB pages 100
4MB pages 25
16MB pages 0
Node[1]
Totalmem 506.5M
freemem 401.7M
64k page 1.5K
256k pages 405
1MB pages 101
4MB pages 25
16MB pages 0
Node[2]
Totalmem 506.5M
freemem 448.6M
64k page 1.6K
256k pages 405
1MB pages 101
4MB pages 25
16MB pages 0
Node[3]
Totalmem 506.5M
freemem 176.2M
Use the "h,j,k,l" keys to move around
Use "D" to delete a section
Use "I" to expand a section

SAR (System Activity Reporter)

Options to SAR

–u	(cpu)
–b	(buffer)
–d	(disks)
–w	(swap & switch)
–q	(run queues)
–p	(paging)
–r	(virtual memory)
–I	(interrupts)
(etc...)	
–A	(everything)

```
sar -r 1 3

IRIX 5.1.1.1 IP20 01/04/94

10:34:53 freemem freeswp
10:34:54     323   36752
10:34:56     320   36752
10:34:57     321   36752


10:34:57 freemem freeswp
Average      321   36752
```

Timex

```
timex -s date
Tue Jan  4 10:39:59 MET 1994

real         0.07
user         0.01
sys          0.03


IRIX dope 5.1.1.1 10011611 IP20    01/04/94

10:39:59  %usr  %sys %intr  %wio %idle %sbrk  %wfs %wswp %wphy %wgsw %wfif
10:40:00    10    70     0    10    10     0   100     0     0     0     0

10:39:59  device   %busy  avque  r+w/s  blks/s    w/s wblks/s  avwait  avserv
10:40:00   dks0d1     30    1.0     15     360      5     160     0.0    20.0
           dks0d2      0    0.0      0       0      0       0     0.0     0.0


(etc...)
```

The SpeedShop Profiler

Profiling Tools

`perfex`: first–pass detection of program–wide performance issues

What is the problem?

SpeedShop: source–procedure annotation for focused experiments

Which procedures have the problems?

`dprof`: virtual memory data space profiling to discover access patterns

Which arrays are involved

SpeedShop

- **Speedshop provides the following profiling experiments:**
  - *PC sampling*
  - *Ideal time*
  - *Usertime*
  - *Hardware counter profiling*
  - *Floating point exception tracing*
  - *Heap tracing*
- **Does not require any special compilation or re–linking**
- **Usage:**

```
% ssrun -exp experiment [options] program [flags]
% prof "ssrun_output_file(s)"
```

- **Modes of counting on some experiments:**
  - **f for more fine–grained sampling**
  - **x for a 32 bit bin size, instead of the default of 16**

Example: `% ssrun -exp [f]pcsamp[x] ./a.out`

PC–sampling experiment

```
% ssrun -pcsamp adi2

 Time:       7.619 seconds
 Checksum:   5.6160428338E+06

% prof adi2.pcsamp.4885

------------------------------------------------------------------------
Profile listing generated Sat Jan  4 10:28:11 1997
    with:       prof adi2.pcsamp.4885
------------------------------------------------------------------------

samples   time     CPU     FPU    Clock    N-cpu  S-interval Countsize
   8574   8.6s R10000 R10010 196.0MHz    1       1.0ms      2(bytes)

Each sample covers 4 bytes for every  1.0ms ( 0.01% of 8.5740s)

------------------------------------------------------------------------
  -p[rocedures] using pc-sampling.
  Sorted in descending order by the number of samples in each procedure.
  Unexecuted procedures are excluded.
------------------------------------------------------------------------

samples   time(%)       cum time(%)       procedure (dso:file)

   6688    6.7s( 78.0)   6.7s( 78.0)          zsweep (adi2:adi2.f)
    671   0.67s(  7.8)   7.4s( 85.8)          xsweep (adi2:adi2.f)
    662   0.66s(  7.7)     8s( 93.6)          ysweep (adi2:adi2.f)
    208   0.21s(  2.4)   8.2s( 96.0)        fake_adi (adi2:adi2.f)
    178   0.18s(  2.1)   8.4s( 98.1)          irand_ (/usr/lib32/libftn.so:../../libF77/rand_.c)
    166   0.17s(  1.9)   8.6s(100.0)           rand_ (/usr/lib32/libftn.so:../../libF77/rand_.c)
      1 0.001s(  0.0)   8.6s(100.0)       __oserror (/usr/lib32/libc.so.1:oserror.c)

   8574   8.6s(100.0)   8.6s(100.0)           TOTAL
```

Ideal Time experiment

For ideal time, speedshop must first "pixify" the executable and DSOs:

```
% ssrun -ideal adi2

Beginning libraries
        /usr/lib32/libssrt.so
        /usr/lib32/libss.so
        /usr/lib32/libfastm.so
        /usr/lib32/libftn.so
        /usr/lib32/libm.so
        /usr/lib32/libc.so.1
Ending libraries, beginning "adi2"
 Time:       8.291 seconds
 Checksum:  5.6160428338E+06
```

Ideal Time experiment (continued)

```
% prof adi2.ideal.4920

  Prof run at: Sat Jan  4 10:34:06 1997
 Command line: prof adi2.ideal.4920


          285898739: Total number of cycles
           1.45867s: Total execution time
          285898739: Total number of instructions executed
              1.000: Ratio of cycles / instruction
                196: Clock rate in MHz
             R10000: Target processor modelled


------------------------------------------------------
Procedures sorted in descending order of cycles executed.
Unexecuted procedures are not listed. Procedures
beginning with *DF* are dummy functions and represent
init, fini and stub sections.
------------------------------------------------------

        cycles(%)  cum %     secs    instrns    calls procedure(dso:file)

  68026368(23.79)  23.79     0.35   68026368 32768 xsweep(adi2.pixie:adi2.f)
  68026368(23.79)  47.59     0.35   68026368 32768 ysweep(adi2.pixie:adi2.f)
  68026368(23.79)  71.38     0.35   68026368 32768 zsweep(adi2.pixie:adi2.f)
  35651584(12.47)  83.85     0.18   35651584 2097152 rand_(./libftn.so.pixn32:../../libF77/rand_.c)
  27262976( 9.54)  93.39     0.14   27262976 2097152 irand_(./libftn.so.pixn32:../../libF77/rand_.c)
  18874113( 6.60)  99.99     0.10   18874113     1 fake_adi(adi2.pixie:adi2.f)
     11508( 0.00)  99.99     0.00      11508     5 memset(./libc.so.1.pixn32:/slayer_xlv0/ficussg-no
      3101( 0.00)  99.99     0.00       3101    55 __flsbuf(./libc.so.1.pixn32:_flsbuf.c)
```

Operation Counts

```
% prof -op adi2.ideal.4920

  Prof run at: Wed Jan 15 14:42:54 1997
 Command line: prof -op adi2.ideal.4920


          285898739: Total number of cycles
           1.45867s: Total execution time
          285898739: Total number of instructions executed
              1.000: Ratio of cycles / instruction
                196: Clock rate in MHz
             R10000: Target processor modelled


------------------------------------------------------------
pixstats summary
------------------------------------------------------------
           56590456: Floating point operations (38.796 Mflops @ 196 MHz)
          105500230: Integer operations (72.3265 M intops @ 196 MHz)

                                .
                                .
                                .
```

Gprof

```
% prof -gprof adi2.ideal.4920

                                        .
                                        .
                                        .
          All times are in milliseconds.
-------------------------------------------------------------------------------
          NOTE: any functions which are not part of the call
                graph are listed at the end of the gprof listing
-------------------------------------------------------------------------------

                                    self              kids        called/total    parents
 index          cycles(%)          self(%)           kids(%)  called+self     name     index
                                    self              kids        called/total    children

   [1]  285895481(100.00%)          57( 0.00%)  285895424(100.00%)     0       __start [1]
                                       50          285895369         1/1         main [2]
                                        3                  0         1/1         __istart [112]
                                        2                  0         1/1         __readenv_sigfpe [113]

-------------------------------------------------------------------------------

                                       50          285895369         1/1         __start [1]
   [2]  285895419(100.00%)          50( 0.00%)  285895369(100.00%)     1       main [2]
                                 18874113          267020445         1/1         fake_adi [3]
                                      205                606         5/5         signal [44]

-------------------------------------------------------------------------------

                                 18874113          267020445         1/1         main [2]
   [3]  285894558(100.00%)    18874113( 6.60%)  267020445(93.40%)     1       fake_adi [3]
                                 68026368                  0     32768/32768     zsweep [4]
                                 68026368                  0     32768/32768     ysweep [5]
                                 68026368                  0     32768/32768     xsweep [6]
                                 35651584           27262976     2097152/2097152     rand_ [7]
                                       28              13486         2/2         s_wsfe64 [9]
                                       26               5368         2/2         e_wsfe [17]
                                       22               2610         1/1         do_fioxr4v [25]
                                       22               2610         1/1         do_fioxr8v [24]
                                       23               2428         1/1         s_stop [28]
                                      114                 44         2/2         dtime_ [68]

-------------------------------------------------------------------------------
                                        .
                                        .
                                        .
```

Usertime Experiment

```
% ssrun -usertime adi2

% prof adi2.usertime.19572
------------------------------------------------------------------------
Profile listing generated Wed Jan 15 16:57:10 1997
    with:       prof adi2.usertime.19572
------------------------------------------------------------------------

         Total Time (secs)      : 9.99
         Total Samples          : 333
         Stack backtrace failed: 0
         Sample interval (ms)   : 30
         CPU                    : R10000
         FPU                    : R10010
         Clock                  : 196.0MHz
         Number of CPUs         : 1


------------------------------------------------------------------------

index   %Samples      self descendents  total         name
[1]     100.0%    0.00         9.99     333           __start
[2]     100.0%    0.00         9.99     333           main
[3]     100.0%    0.09         9.90     333           fake_adi
[4]      80.8%    8.07         0.00     269           zsweep
[5]       7.5%    0.75         0.00      25           xsweep
[6]       6.9%    0.69         0.00      23           ysweep
[7]       3.9%    0.12         0.27      13           rand_
[8]       2.7%    0.27         0.00       9           irand_
```

Hardware Counter Profiling

The most commonly used counters are named:

[f]gi_hwc:	graduated instructions
[f]cy_hwc:	cycle counter
[f]ic_hwc:	primary icache misses
[f]isc_hwc:	secondary icache misses
[f]dc_hwc:	primary dcache misses
[f]dsc_hwc:	secondary dcache misses
[f]tlb_hwc:	TLB misses
[f]gfp_hwc:	graduated floating point instructions

Can use any counter as follows:

prof_hwc: get counter from environment variable

```
% setenv _SPEEDSHOP_HWC_COUNTER_NUMBER <cntr>
% setenv _SPEEDSHOP_HWC_COUNTER_OVERFLOW <ovfl>
```

Hardware Counter Profiling (cont'd)

```
% ssrun -tlb_hwc adi2
 Time:      7.887 seconds
 Checksum:  5.6160428338E+06
% prof adi2.tlb_hwc.18520
------------------------------------------------------------------------
Profile listing generated Wed Jan 15 19:27:52 1997
    with:       prof adi2.tlb_hwc.18520
------------------------------------------------------------------------

        Counter                 : TLB misses
        Counter overflow value: 257
        Total numer of ovfls    : 21181
        CPU                     : R10000
        FPU                     : R10010
        Clock                   : 195.0MHz
        Number of CPUs          : 1


------------------------------------------------------------------------
  -p[rocedures] using counter overflow.
  Sorted in descending order by the number of overflows in each procedure.
  Unexecuted procedures are excluded.
------------------------------------------------------------------------

  overflows(%)  cum overflows(%)      procedure (dso:file)

  21169( 99.9)      21169( 99.9)          zsweep (adi2:adi2.f)
      4(  0.0)      21173(100.0)          ysweep (adi2:adi2.f)
      4(  0.0)      21177(100.0)          xsweep (adi2:adi2.f)
      4(  0.0)      21181(100.0)       fake_adi (adi2:adi2.f)

  21181(100.0)                        TOTAL
```

Floating Point Exceptions

Trace with SpeedShop's –fpe experiment

Excessive underflows can cause large system time

Use libfpe to flush them to zero

```
% ld -o program ... -lfpe
% setenv TRAP_FPE "UNDERFL=ZERO"
% program ...
```

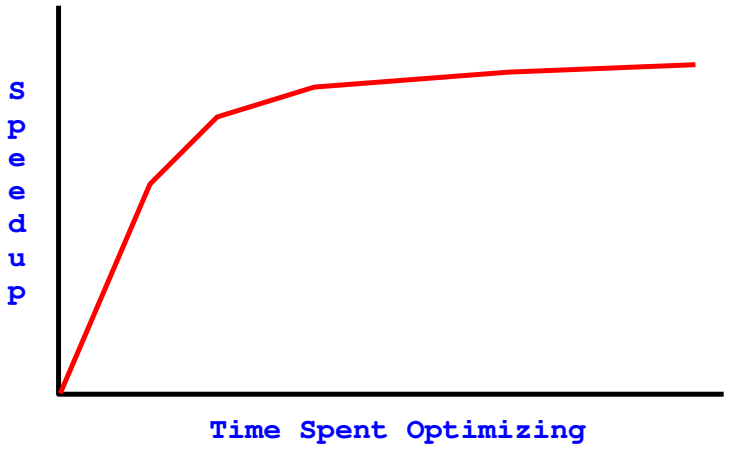
```
#include <sys/fpu.h>

void
flush_to_zero(int on_off)
{
union fpc_csr   n;

     n.fc_word = get_fpc_csr();
     if ( on_off == 0 ) {
          n.fc_struct.flush = 0;
     } else {
          n.fc_struct.flush = 1;
     }
     set_fpc_csr(n.fc_word);
}
```

See handle_sigfpes(3C)

Profiling

Optimization Methodology

- Optimize one loop/routine at a time
- Start with the most time consuming routines
- Then the second most ...
- Then the third most ...
- Parallelize your program



Performance Boosters

Feedback Compilation

Provides run–time execution frequencies for better compiler optimization decisions

1. Compile the program –O0
2. "Pixify" the program: `pixie foo`
3. Run the `.pixie` executable using "representative" input data

`.Counts` file will be produced
rename `.Counts` file before each run
2 & 3 can be replaced by `ssrun -ideal`

Feedback Compilation (continued)

4 Run prof with –feedback –pixie:

```
% prof -feedback -pixie foo foo.1.counts {foo.2.counts, ...}
```

produces .fb file for cord
.cfb file for feedback compilation

5 Recompile the program:

```
% cc|f77 -c -fb foo.cfb + optimization flags
```

Notes:

The.cfb file can be reused if the source is not changed

Non–representative input data can mislead the compiler

Benefits of Feedback Compilation

Better decisions on:

- ***Inlining***
- ***Register allocation***
- ***Unrolling and software pipelining***
- ***If–conversion***
- ***Loop–invariant code motion***

  (not moved out of loop if executed less than n% of iterations, where n is set by the following compiler option: `-WOPT:aggcm_thresh=n` [default 70])

The Cord Utility

- Reduces instruction cache misses and virtual paging
- Works by rearranging order of procedures in executable or library

  Procedures invoked together are located near each other

- How to use it:

```
% cord foo.exe foo.fb {foo.2.fb, ...}
```

One can use multiple files to provide temporal proximity information to cord, e.g., in a phase–structured program, one .fb file for each phase