

ORACLE®



ORACLE[®]

Drilling Down into Performance Data with the Oracle Solaris Studio Performance Analyzer

Marty Itzkowitz marty.itzkowitz@oracle.com

Yukon Maruyama yukon.maruyama@oracle.com

CScADS, July 16, 2013

Agenda

- The Problem
- Filtering and Navigation
 - Functions, source, disassembly
 - PCs, Lines
 - Callers-callees
 - Timeline
- Demo

The Problem

- Extract the signal from the noise
- Complex runs
 - Many processes, threads, CPUs
 - Processes interact in many ways
 - And affect each other's performance
 - Most activity is uninteresting
- Filter the data to decrease noise
- Navigate through the data to explore the real signal

Minimize the number of mouse clicks to do it

Filtering and Navigation

- Filtering: slice and dice the data in many ways
 - By function, source line, or instruction in stack or not in stack
 - By callstack fragment
 - By time
 - By cache line, page, address, LWP, thread, CPU, ...
 - And any combinations of these properties
- Navigation from anywhere to anywhere
 - From function to source, disassembly, callers, callees, ...
 - From Timeline event to callstack to source, disassembly, ...
- Filtering and navigation accessible from option menu in View

From the Function List

Excl. Total CPU (sec.)	Incl. Total CPU (sec.)	Name
23.166	23.166	<Total>
6.064	6.064	real_recurse
3.102	3.102	gpf_work
3.012	3.012	muldiv
3.002	3.002	cputime
2.992	2.992	icputime
2.332	2.332	my_irand
0.700	3.032	fitos
0.560	0.560	dousleep
0.510	0.510	inc_middle
0.290	0.290	inc_entry
0.090	0.090	inline_code
0.090	0.090	macro_code
0.080	0.080	s_inline_co
0.070	0.070	ext_macro_co
0.060	0.060	inc_body
0.060	0.060	inc_brace
0.060	0.060	inc_exit

Show Source

Show Disassembly

Add Filter: Include only stacks containing the selected functions

Add Filter: Include only stacks not containing the selected functions

Add Filter: Include only stacks with the selected functions as leaf

Add Filter: Include only stacks containing similarly-named functions

Add Filter: Advanced Custom Filter...

Undo Last Filter

Redo Filter

Remove All Filters

Sort by

Properties

Navigate to source, disassembly; apply several varieties of filters

From the Source View, I

```
1067.     int    i;
1068.     int    imax;
1069.     volatile float x= 0.0;
1070.
1071.     imax = 4* amt * amt;
1072.
1073.     for(i = 0; i < imax;
1074.         //volatile f
1075.         int j;
1076.         x = 0.0;
1077.         for(j=0; j<2
1078.             x =
1079.         }
1080.     }
1081.     return x;
```

Recent St

- Back
- Forward
- Show Callee Source
- Show Callee Disassembly
- Show Caller Source
- Show Caller Disassembly
- Show Disassembly
- Add Filter: Include only stacks containing the selected lines
- Add Filter: Include only stacks not containing the selected lines
- Add Filter: Advanced Custom Filter...
- Undo Last Filter
- Redo Filter
- Remove All Filters
- Previous Hot Line ^-P
- Next Hot Line ^-N
- Previous Non-zero Metric Line ^+⇧-P
- Next Non-zero Metric Line ^+⇧-N
- Properties

Calls

f_work called by

f_b

f_a

For a line with no calls in it

From the Source View, II

The screenshot shows a debugger's source view with a C function call highlighted. The code is as follows:

```
374.         pvstart = gethrvtime();
20.094 375.         (k->function)(k->param);
376.         pend = gethrtime();
377.         pvend = gethrvtime();
378.         fprintf(fid,
379.             "%c %6.3f %6.3f %s\n",
380.             (k->noverify == 0? 'X' : 'Y'),
381.             (double)(pend - pstart)/(double)10,
382.             (double)(pvend - pvstart)/(double)10,
383.             k->acctname);
384.         fflush(fid);
385. #if OS(Solaris)
386.         /* verify the signal mask
387.         check_sigmask();
388. #endif /* OS(Solaris) */
389.
390.         break;
391.     }
392. }
393.
394. if(k->name == NULL) {
395.     sprintf(buf, "++ ignoring `%s'\n",
396.         fprintf(stderr, buf);
397. }
398.
```

A context menu is open over the highlighted line 375. The menu items are:

- Back
- Forward
- Show Callee Source
- Show Callee Disassembly
- Show Caller Source
- Show Caller Disassembly
- Show Disassembly
- Add Filter: Include only stacks containing the selected lines
- Add Filter: Include only stacks not containing the selected lines
- Add Filter: Advanced Custom Filter...
- Undo Last Filter
- Redo Filter
- Remove All Filters
- Previous Hot Line (^-P)
- Next Hot Line (^-N)
- Previous Non-zero Metric Line (^+^P)
- Next Non-zero Metric Line (^+^N)
- Properties

On the right side of the menu, a list of call targets is shown:

- Inclusive
- 556 (87.98%)
- 094 (86.74%)
- 094 (86.74%)
- cputime
- dousleep
- endcases
- fitos
- gpf
- icputime
- multdiv
- recurse

From a line with a call to multiple functions, choice of callees

From the Disassembly View, I

The screenshot displays a disassembler window with assembly code on the left and a context menu on the right. The assembly code is organized into blocks: block 375 (addresses 406e66-406e77), block 376 (addresses 406e7a-406e87), block 377 (addresses 406e8e-406e9b), and block 378 (addresses 406ea2-406ea6). The instruction at address 406e77 is highlighted in yellow. The context menu is open over this instruction, listing various navigation and filtering options.

Address	Disassembly
U. [375] 406e66:	movq -0x30(%rbp),%r8
0. [375] 406e6a:	movl 0x18(%r8),%edi
0. [375] 406e6e:	movq 8(%r9),%r8
0. [375] 406e72:	movl \$0,%eax
20.094 [375] 406e77:	call *%r8d
376. pend = geth	
0. [376] 406e7a:	movl \$0,%eax
0. [376] 406e7f:	call gethrtime [0x405
0. [376] 406e84:	movq %rax,%r8
0. [376] 406e87:	movq %r8,-0x448(%rbp)
377. pvend = geth	
0. [377] 406e8e:	movl \$0,%eax
0. [377] 406e93:	call gethrvtime [0x405
0. [377] 406e98:	movq %rax,%r8
0. [377] 406e9b:	movq %r8,-0x450(%rbp)
378. fprintf(fid	
379.	"%c %6.3f
380.	(k->noverify == 0?
381.	(double)(pend - pst
382.	(double)(pvend - pvs
383.	k->acctname);
0. [383] 406ea2:	movq -0x30(%rbp),%r8
0. [383] 406ea6:	movl 0x10(%r8),%eax

Source File: ...
Object File: ...

- Back
- Forward
- Show Callee Source
- Show Callee Disassembly
- Show Caller Source
- Show Caller Disassembly
- Add Filter: Include only stacks containing the selected PCs
- Add Filter: Include only stacks not containing the selected PCs
- Add Filter: Advanced Custom Filter...
- Undo Last Filter
- Redo Filter
- Remove All Filters
- Previous Hot Line ^-P
- Next Hot Line ^-N
- Previous Non-zero Metric Line ^+⬆-P
- Next Non-zero Metric Line ^+⬆-N
- Properties

From a non-call instruction

From the Disassembly View, II

The screenshot displays a disassembler interface with assembly code on the left and a context menu on the right. The assembly code includes instructions like `movq -0x30(%rbp), %r9`, `movq -0x30(%rbp), %r8`, `movl 0x18(%r8), %edi`, `movq 8(%r9), %r8`, `movl $0, %eax`, `call *%r8d`, `movl $0, %eax`, `call gethrtime`, `movq %rax, %r8`, `movq %r8, -0x4(%r8)`, `movl $0, %eax`, `call gethrtime`, `movq %rax, %r8`, `movq %r8, -0x4(%r8)`, `movq -0x30(%rbp), %r9`, `movl 0x1c(%r8), %eax`, and `cmpl $0, %eax`. A context menu is open over the `call *%r8d` instruction, listing options such as `Back`, `Forward`, `Show Callee Source`, `Show Callee Disassembly`, `Show Caller Source`, `Show Caller Disassembly`, and various filter and navigation options.

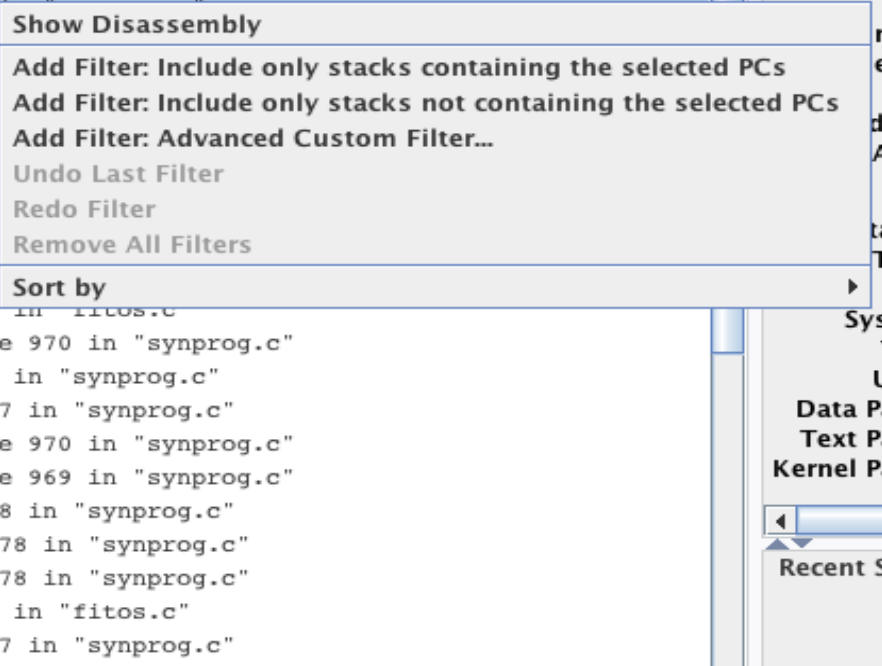
PC Address: 406e77
Size: 3
Source File: /mitzkowi/
Object File: /mitzkow
Load Object: /workspac
Mangled Name:
Aliases:

Back
Forward
Show Callee Source
Show Callee Disassembly
Show Caller Source
Show Caller Disassembly
Add Filter: Include only stacks containing the selected PCs
Add Filter: Include only stacks not containing the selected PCs
Add Filter: Advanced Custom Filter...
Undo Last Filter
Redo Filter
Remove All Filters
Previous Hot Line ^-P
Next Hot Line ^-N
Previous Non-zero Metric Line ^+Q-P
Next Non-zero Metric Line ^+Q-N

From a call instruction; choice of callees

From the PCs Views

23.100	23.100	~10ca17
2.252	2.252	muldiv + 0x0000008B, line 836
1.681	1.681	icputime + 0x0000007D, line 72
1.451	1.451	real_recurse + 0x000000AA, lin
1.421	1.421	real_recurse + 0x000000A6, lin
1.291	1.291	real_recurse + 0x0000009D, lin
0.921	0.921	gpf_work + 0x00000066, line 10
0.831	0.831	cputime + 0x0000007D, line 699
0.791	0.791	gpf_work + 0x0000006A, line 10
0.690	0.690	cputime + 0x00000081, line 699
0.630	0.630	my_irand + 0x00000067, line 31 in fitos.c
0.590	0.590	real_recurse + 0x00000099, line 970 in "synprog.c"
0.560	0.560	cputime + 0x00000074, line 699 in "synprog.c"
0.470	0.470	icputime + 0x0000007A, line 727 in "synprog.c"
0.470	0.470	real_recurse + 0x00000094, line 970 in "synprog.c"
0.470	0.470	real_recurse + 0x000000BB, line 969 in "synprog.c"
0.460	0.460	icputime + 0x00000065, line 728 in "synprog.c"
0.410	0.410	gpf_work + 0x0000005D, line 1078 in "synprog.c"
0.400	0.400	gpf_work + 0x00000059, line 1078 in "synprog.c"
0.400	0.400	my_irand + 0x00000061, line 30 in "fitos.c"
0.380	0.380	icputime + 0x00000074, line 727 in "synprog.c"



The screenshot shows a debugger window with a list of PCs (Program Counter) addresses and their corresponding source code locations. A context menu is open over the selected row, offering options like 'Show Disassembly', 'Add Filter', and 'Sort by'.

PCs should also go to source, callers; callees if a call instruction

From the Lines View

The screenshot displays a debugger's 'Lines View' window. On the left, a list of code lines is shown with columns for PC address, instruction address, and the instruction text. The line 'real_recurse, line 970 in "synprog.c"' is selected. A context menu is open over this line, offering options such as 'Show Source', 'Show Disassembly', and various filtering options. The PC Address field at the top right shows '2:11'.

PC Address	Instruction Address	Instruction Text
23.166	23.166	<Total>
5.224	5.224	real_recurse, line 970 in "synprog.c"
2.802	2.802	muldiv, line 836 in "synprog.c"
2.762	2.762	gpf_work, line 1078 in "synprog.c"
2.612	2.612	cputime, line 699 in "synprog.c"
2.532	2.532	icputime, line 727 in "synprog.c"
0.841	0.841	real_recurse, line 969 in "synprog.c"
0.771	0.771	my_irand, line 31 in "fitos.c"
0.600	0.600	my_irand, line 29 in "fitos.c"
0.530	0.530	my_irand, line 30 in "fitos.c"
0.460	0.460	icputime, line 728 in "synprog.c"
0.430	0.430	dousleep, line 602 in "synprog.c"
0.390	0.390	cputime, line 698 in "synprog.c"
0.350	2.682	fitos, line 54 in "fitos.c"
0.340	0.340	gpf_work, line 1077 in "synprog.c"
0.270	0.270	my_irand, line 21 in "fitos.c"
0.220	0.220	inc_middle, line 207 in "endcases.c"
0.210	0.210	muldiv, line 835 in "synprog.c"
0.200	0.200	inc_entry, line 175 in alternate source context "endcases.c"

Context Menu Options:

- Show Source
- Show Disassembly
- Add Filter: Include only stacks containing the selected lines
- Add Filter: Include only stacks not containing the selected lines
- Add Filter: Advanced Custom Filter...
- Undo Last Filter
- Redo Filter
- Remove All Filters
- Sort by

Lines should also go to callees if a call is on that line

The Callers-Callees View

Callers

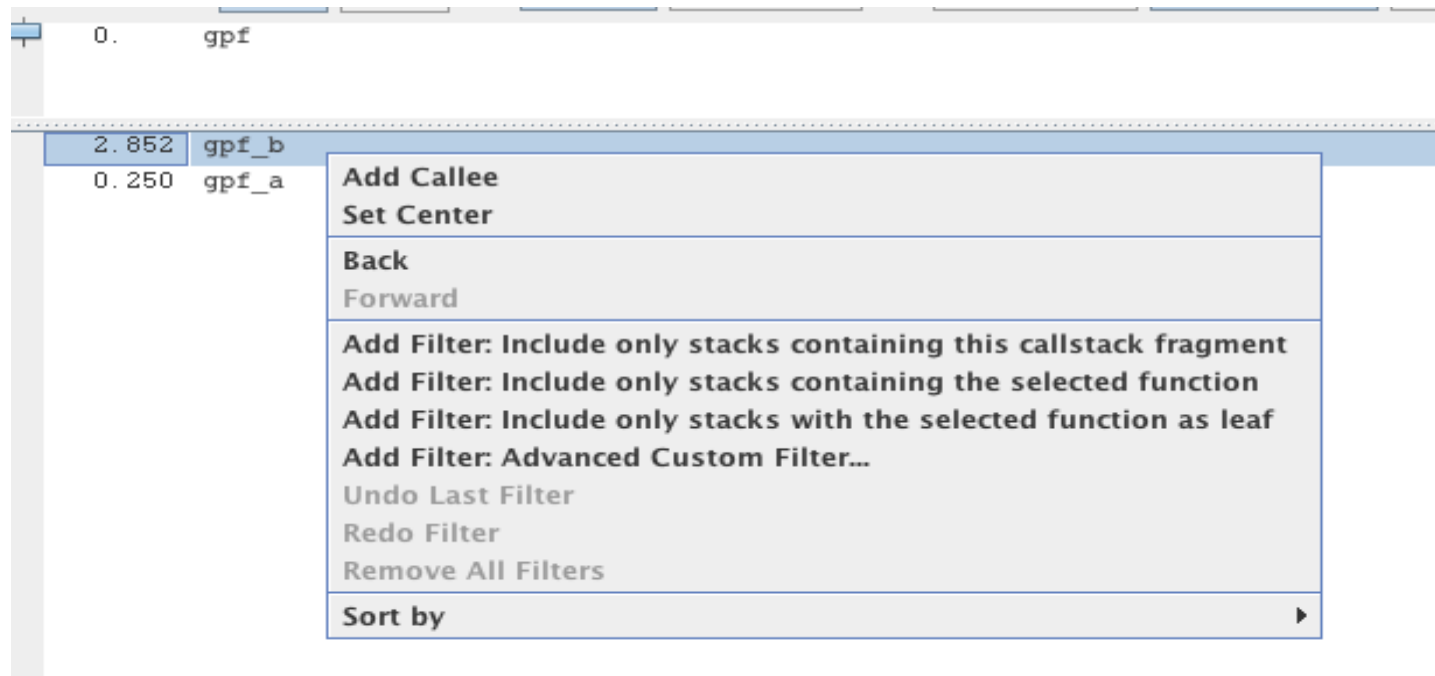
Selected function

Callees

Total CPU (sec.)	Name
3.102	commandline
0.	gpf
2.852	gpf_b
0.250	gpf_a

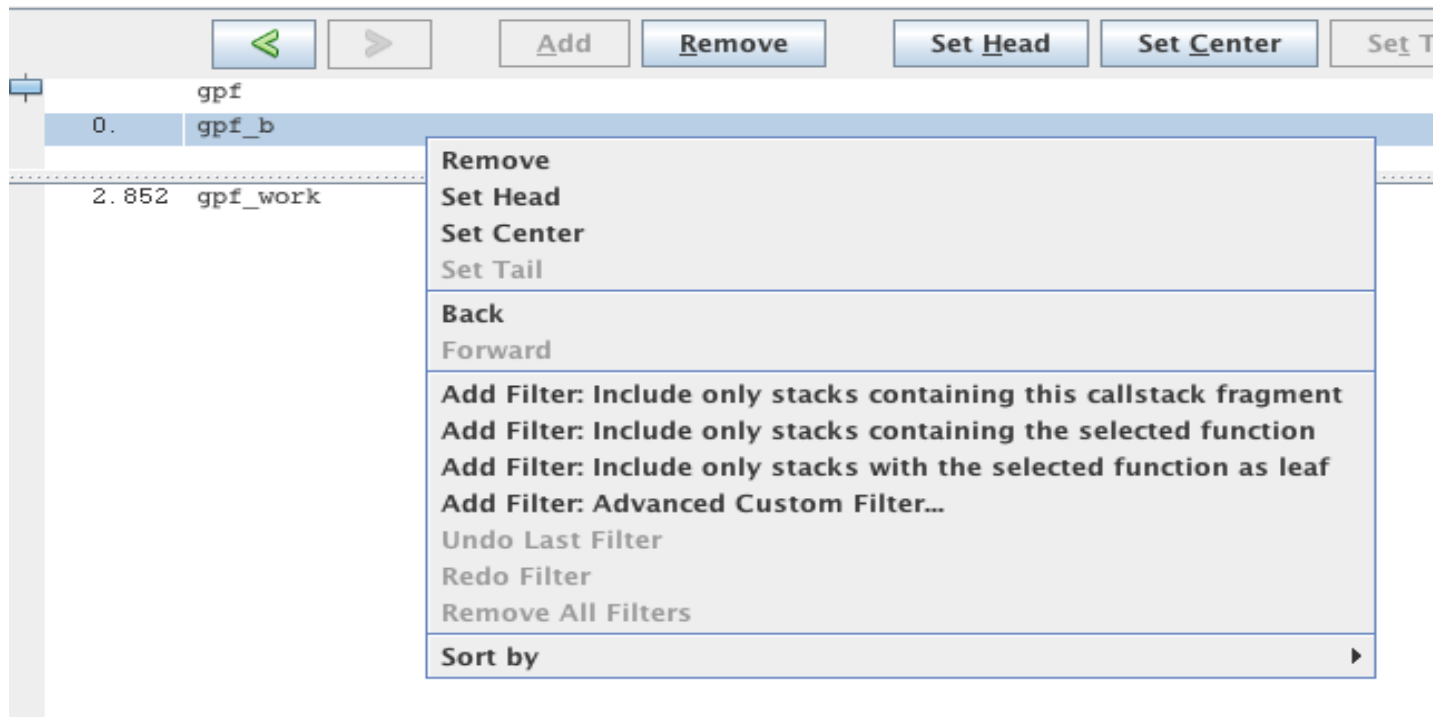
Shows callers and callees of function; can build a callstack fragment

From the Callers-Callees View



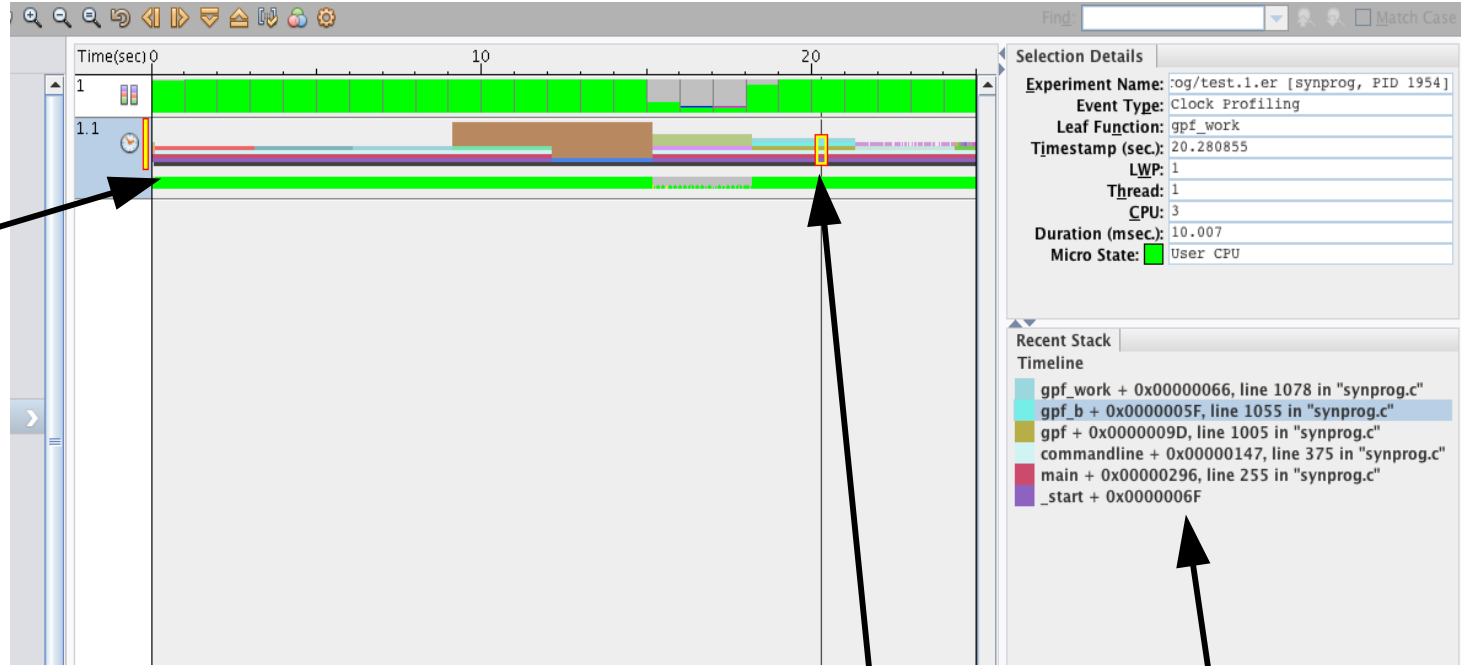
Adding to stack fragment from one of the callees; same for any caller

From the Callers-Callees View



After constructing a callstack fragment: filter by fragment

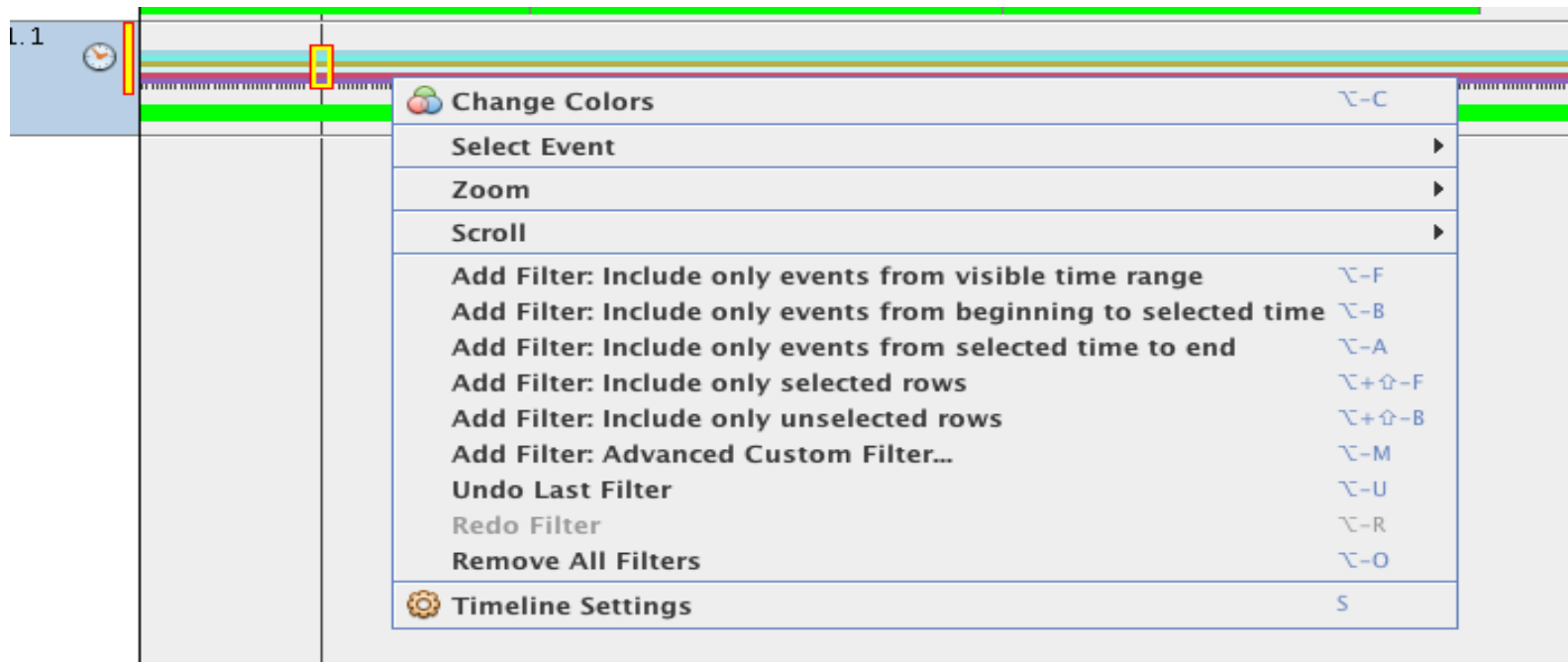
The Timeline



Varying pattern of behavior

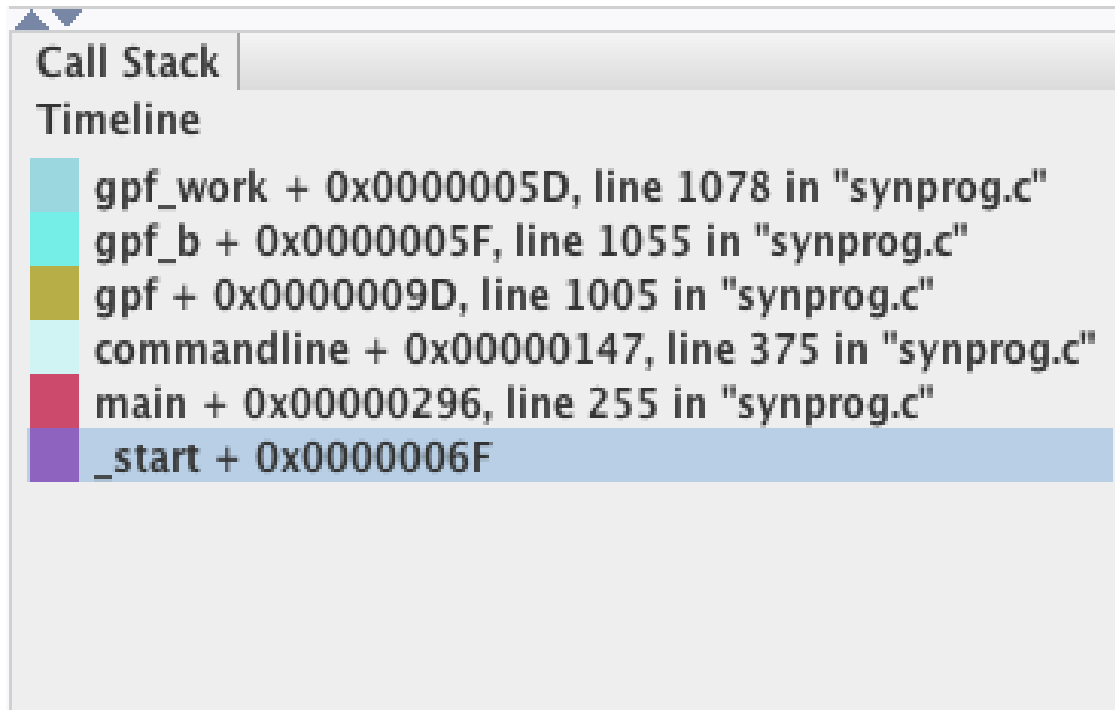
Shows sequence of profile events; select event to see its callstack

Filtering from the Timeline



Filter by time; filter by rows

Navigating from the Timeline



The screenshot shows a debugger window with a 'Call Stack' tab selected. Below the tab, the word 'Timeline' is visible. The call stack contains six frames, each with a colored bar to its left. The frames are listed from top to bottom: 'gpf_work + 0x0000005D, line 1078 in "synprog.c"' (light blue), 'gpf_b + 0x0000005F, line 1055 in "synprog.c"' (cyan), 'gpf + 0x0000009D, line 1005 in "synprog.c"' (olive green), 'commandline + 0x00000147, line 375 in "synprog.c"' (light cyan), 'main + 0x00000296, line 255 in "synprog.c"' (maroon), and '_start + 0x0000006F' (purple). The bottom frame, '_start + 0x0000006F', is highlighted with a light blue background.

```
Call Stack  
Timeline  
gpf_work + 0x0000005D, line 1078 in "synprog.c"  
gpf_b + 0x0000005F, line 1055 in "synprog.c"  
gpf + 0x0000009D, line 1005 in "synprog.c"  
commandline + 0x00000147, line 375 in "synprog.c"  
main + 0x00000296, line 255 in "synprog.c"  
_start + 0x0000006F
```

Stack of selected event; navigate to any frame

Index and Memory Objects

- Defined by formula mapping event to an integer index
 - Some are pre-defined (threads, CPUs, seconds)
 - Others are user-defined
- Index Objects -- apply to all data
 - `indxobj_define KLWP "LWPID"`
 - Aggregate by kernel thread (known as LightWeight Process)
- Memory Objects -- apply to dataspace profile data only
 - `mobj_define Vline_32B " ((VADDR>255) ? (VADDR>> 5) : -1) "`
 - Index of 32-byte cache line

Filtering from IndexObject

<input type="checkbox"/> Excl. Total CPU ▼ (sec.)	Name
23.166	<Total>
1.001	Second of execution 1
1.001	Second of execution 2
1.001	Second of execution 3
1.001	Second of execution 5
1.001	Second of execution 6
1.001	Second of execution 7
1.001	Second of execution 8
1.001	Second of execution 9
1.001	Second of execution 10
1.001	Second of execution 11
1.001	Second of execution 12
1.001	Second of execution 13
...	...

Add Filter: Include only events with selected items

Add Filter: Include only events without selected items

Add Filter: Advanced Custom Filter...

Undo Last Filter

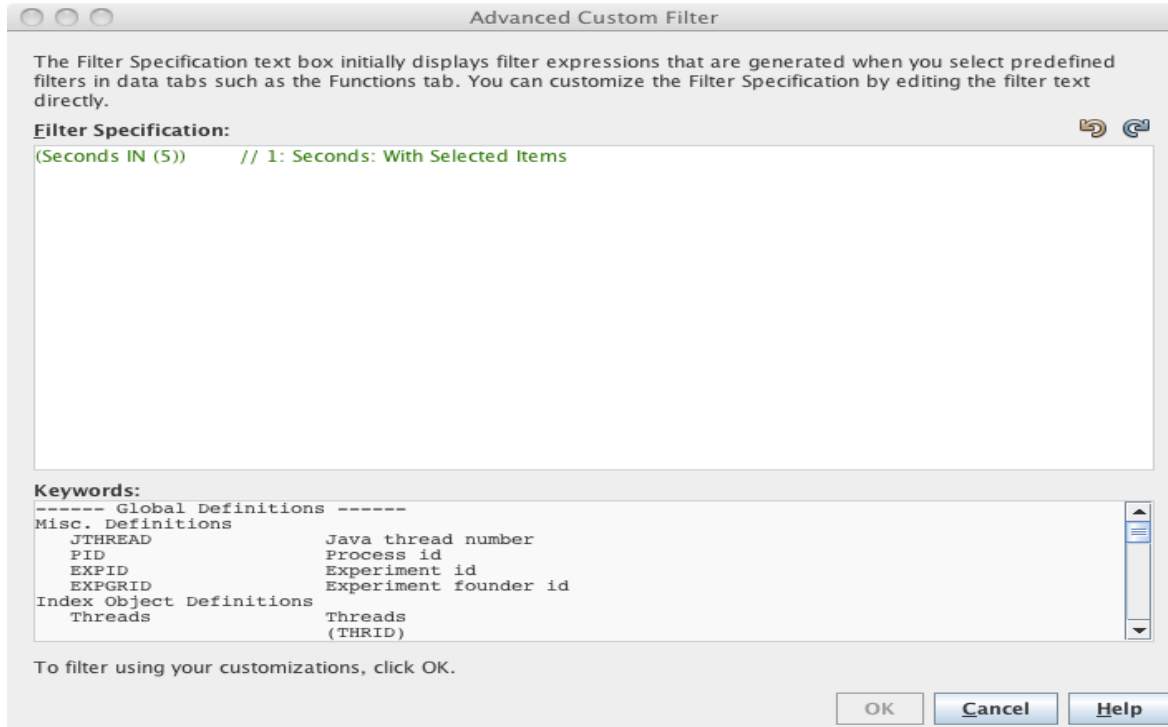
Redo Filter

Remove All Filters

Sort by ▶

In this case, seconds of execution

Advanced Filtering



See existing filter clauses; edit, delete, or add clauses

Demo

- Experiment on a 24-CPU Solaris 11 Nehalem system
- `er_kernel` command
 - Kernel experiment, with user subexperiments
 - One on `dd`; one on `ksynprog`; one on `er_kernel` itself
- Clock-profiling
 - Kernel CPU time in kernel experiment
 - User and System CPU time in user subexperiments
- Explore various filters, Views, and navigating among them



ORACLE[®]

Drilling Down into Performance Data with the Oracle Solaris Studio Performance Analyzer

Marty Itzkowitz marty.itzkowitz@oracle.com

Yukon Maruyama yukon.maruyama@oracle.com

CScADS, July 16, 2013

ORACLE®