



HP-UX 11i Knowledge-on-Demand

HP technical Webcast series: software optimization



Technology for better business outcomes

HP-UX 11i v3 Knowledge-on-Demand

- Objective: Support software development partners and customers in achieving better business outcomes with HP-UX 11i.
- What HP is providing: a series of technical on-demand training Webcasts
 - Focused on helping developers increase performance through application optimization for HP-UX 11i v3 on HP Integrity servers
 - Access to HP for follow-up questions
 - Available at www.hp.com/go/knowledgeondemand

HP-UX 11i v3 Knowledge-on-Demand Webinars – planned curriculum

- Foundation Track
 - Module 1: How to upgrade to HP-UX 11i v3
 - Module 2: HP-UX open source resources
 - Module 3: Unified file cache
 - Module 4: Caliper
 - Module 5: NUMA Tuning: Getting the Most Out of Your Cellular Server by using NUMA
 - Module 6: The Mercury Library – Increasing Application Performance
 - Module 7: Software Transition Kit's (STK's) for HP-UX 11i v3
- Java Developers Track
 - Module 8: Java Memory Management - Internals and Performance
 - Module 9: HPjmeter – measure Java application performance on HP-UX 11i
 - Module 10: Solving Java performance problems
- C/C++ Developers Track
 - Module 11: pthreads enhancements in HP-UX 11i v3
 - Module 12: Kernel tracing & profiling tools (internal tools)
 - Module 13: Using compilers to get optimal performance
 - Module 14: HP Code Advisor: A Powerful New C/C++ Analysis Tool for HP-UX
 - Module 15: Montecito Hyper-Threading on HP-UX 11i v3

Additional Webinars
published going forward!

Related HP-UX 11i v3 resources

- All developers' resources
 - HP-UX 11i developers' content
www.hp.com/go/hpuxdev
 - HP-UX 11i v3 news, functionality, product download and services resources
www.hp.com/go/hpux11i
 - HP Integrity server ISV resources for DSPP members
www.hp.com/go/dspp_integrity
 - HP Integrity server product information
www.hp.com/go/integrity
- Software partner promotional opportunity
 - HP promotion for HP-UX 11i v3-ready software partner application
www.hp.com/go/v3promotion

Enjoy this Knowledge-on-Demand topic!

Thank you for taking time to learn about HP-UX 11i v3 and related technologies.

Please send comments on today's topic and/or requests for future topics to:

hpuxquestions@hp.com



HP Caliper performance tool for Integrity servers: New features for understanding performance

An HP-UX 11i Knowledge-on-Demand software optimization Webcast

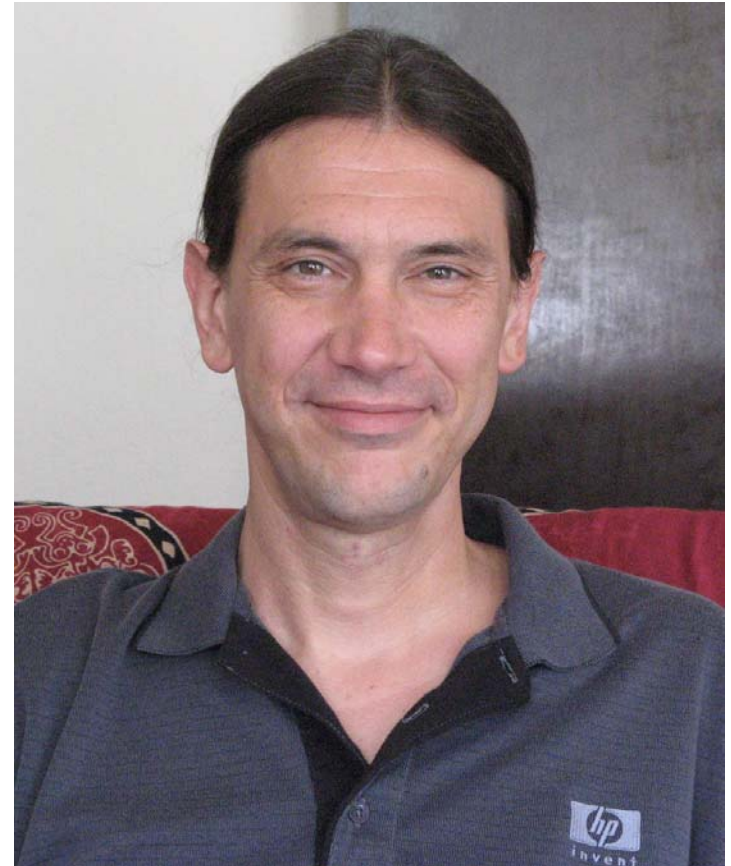


Technology for better business outcomes

Introducing today's speaker

Stephen Williams is a member of the HP Caliper team and has worked at HP for the past 18 years.

He has worked on debuggers and performance tools and has specialized in user interfaces.



Agenda

- Quick overview of HP Caliper
- New features in HP Caliper 4.2 and 4.3
- New graphical user interface (GUI) features
- Future directions
- Summary

What is HP Caliper?

- Per-process or system-wide performance measurement tool, for any Intel® Itanium®/Itanium 2 processor native application
- For Integrity servers running HP-UX and Linux
- “Swiss Army knife”
 - Many different measurements
 - Common user interface and options
 - Multiple report formats w/source and assembly
- GUI and command-line tool
- Uses Performance Monitor Unit (PMU) hardware and dynamic instrumentation as needed
- Measures any application without special builds
- Per-process and system-wide measurements



Measured applications

- HP, Intel[®], and GNU compilers
- C, C++, Fortran9x, assembly, Java
- 32-bit, 64-bit executable
- Single-process, multi-process
- Single-thread, multi-threaded
- Optimized, unoptimized, debug, stripped
- Main, static/dynamic shared libraries
- Entire run, partial run, server/daemon
- PA-RISC binaries on HP-UX tracked

“Just run it.”

Measurements

	Measurement	Use
Overview:	cpu, ecount	What?
Profiles:	alat, branch, dcache, dtlb, fprof, icache, itlb, cycles, traps	Where?
Traces:	pmu_trace	
Call stack:	cstack	
Add-on:	--memory-usage*, --system-usage* --data-summary	
Call graph:	scgprof, cgprof*	Details?
Coverage:	fcover*	(instrumented)
Counts:	acount*, fcount*	

*Not in Linux version

Command line examples

```
caliper [measurement] [options] application [app-opts ]  
caliper [measurement] [options] PID1 [PID2 ...]  
caliper [measurement] [options] -w
```

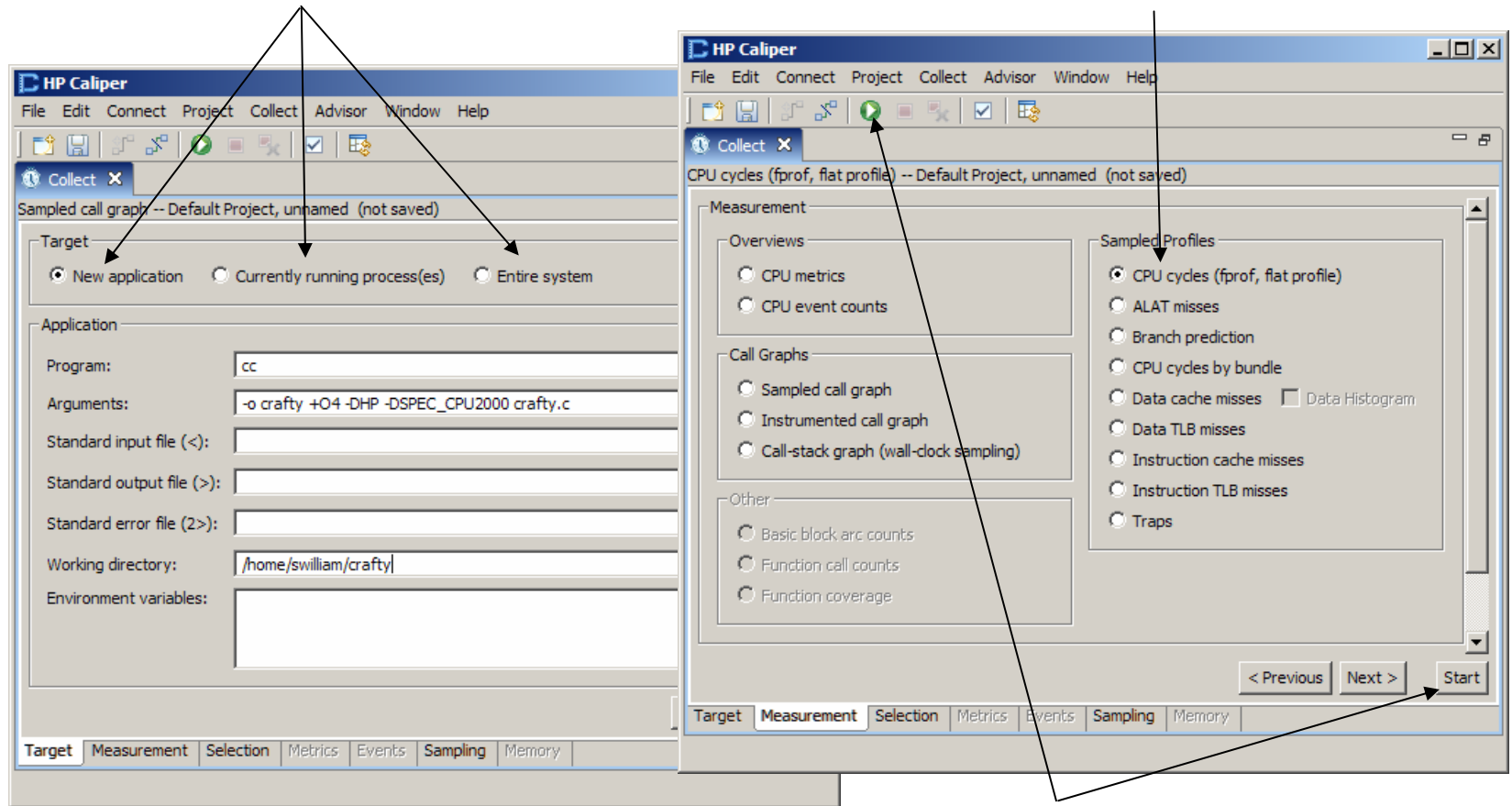
Examples:

```
caliper fprof -o out.txt sweep3d  
caliper dcache -t -p all cc himom.c  
caliper cpu -w -o out.txt --dur 10  
caliper scgprof -p myproc myscript.sh  
caliper icache -o out.txt 8451 8452 8453
```

Collecting data using Caliper GUI

Choose target

Choose measurement



Start collection

Text reports (overview section)

CPU data
(all processes)

```
=====
HP Caliper A.4.3.108 Report Summary for Flat Profile
=====
```

```
-----
Collection Run 1 (Flat Profile)
-----
```

Processor Information

```
Machine name:      fitzroy
Number of processors: 4
Processor type:    Itanium2 9M
Processor speed:   1600 MHz
Virtual machine:   no
```

Run Information

```
Configuration:    /opt/caliper/config/fprof
Date:             July 19, 2007
Version:          HP Caliper - HP-UX Itanium Version A.4.3.108
OS:              HP-UX B.11.23 U ia64
Database:         /home/swilliam/sw.db
Measurement scope: per-process
```

Sampling Specification

```
Sampling event:   CPU_CYCLES
Sampling period:  500000 events
Sampling period variation: 25000 (5.00% of sampling period)
Sampling counter privilege: user (user-space sampling)
Data granularity: 16 bytes
Data sampled:     IP
```

Run
information
(all processes)

Metrics Summed for Entire Run

```
-----
PLM
Event Name      U..K TH      Count
-----
CPU_CYCLES      x___ 0      88037986979
BACK_END_BUBBLE.ALL  x___ 0      34035683649
BE_EXE_BUBBLE.GRALL x___ 0      13006566279
-----
```

PLM: Privilege Level Mask

```
U/K = user/kernel levels (U: level 3, K: level 0)
The intermediate levels (1, 2) are unused on HP-UX or Linux
x : the metric is measured at the given level (_ : not measured)
TH: event THreshold, determines the event counter behavior,
TH == 0 : counter += event_count_in_cycle
TH > 0 : counter += (event_count_in_cycle >= threshold ? 1 : 0)
```

% Cycles lost due to stalls (lower is better):

```
38.66 = 100 * (BACK_END_BUBBLE.ALL / CPU_CYCLES)
```

% Cycles stalled due to GR/GR or GR/load dependency (lower is better):

```
14.77 = 100 * (BE_EXE_BUBBLE.GRALL / CPU_CYCLES)
```

Process Summary

```
-----
% Total  Cumulat
   IP    % of      IP
Samples  Total      Samples  Process
-----
89.90    89.90      158312   be (PID: 13207)
 8.73    98.63      15368   ecom (PID: 13196)
 1.28    99.90       2247   u2comp (PID: 13202)
.
.
.
-----
```

Per-process
data

Text reports (per-process sections)

```
=====
Flat Profile Report for be
=====
```

Report for be process

```
Target Application
Program: /opt/langtools/lbin/be
Invocation: /opt/langtools/lbin/be -keep_dir ...
Process ID: 13207 (exec'd by sh, Process ID 13207)
Start time: 02:13:39 PM
End time: 02:14:30 PM
Termination Status: 0
Last modified: March 31, 2006 at 10:35 AM
Memory model: ILP32
Processor set: default
```

```
Target Execution Time
Real time: 51.262 seconds
User time: 50.121 seconds
System time: 0.269 seconds
```

```
Sampling Specification
Number of samples: 158312
Data sampled: IP
```

Metrics Summed for All Processes

CPU data

Event Name	PLM		Count
	U..K	TH	
CPU_CYCLES	x___	0	79142392492
BACK_END_BUBBLE.ALL	x___	0	30955135677
BE_EXE_BUBBLE.GRALL	x___	0	11949987303

Load Module Summary

Per-module data

% Total	Cumulat		
IP	% of	IP	
Samples	Total	Samples	Load Module
78.90	78.90	124905	be
19.14	98.04	30298	libc.so.1
1.95	99.98	3083	libcsup.so.1

Function Summary

Per-function data

% Total	Cumulat			
IP	% of	IP	Function	File
Samples	Total	Samples		
8.12	8.12	12855	libc.so.1::real_malloc	malloc.c
3.80	11.92	6021	be::__milli_rem32U	
2.68	14.61	4249	libc.so.1::real_free	malloc.c

Function Details

Function source/ disassembly

% Total	Line		
IP	IP	Slot	>Statement
Samples	Samples	Col,Offset	Instruction
8.12	[libc.so.1::real_malloc, 0x40f36a0, malloc.c]		
	12855	~2576	Function Totals

GUI reports (run information)

For all processes

For specific process

View Run Summary

The screenshot displays two overlapping windows from the HP Caliper application. The left window, titled 'HP Caliper', shows a 'Run Summary' report for 'All Executables'. The right window, titled 'CPU Cycles Run (fitzroy, 2007-07-21 9:07 AM)', shows a 'Run Summary for be' report for a specific process.

Run Summary (All Executables)

HP Caliper A.4.3.10 Flat Profile Multi-Process Report Summary

Processor Information

Machine name:	fitzroy
Number of processors:	4
Processor type:	Itanium2 9M
Processor speed:	1600 MHz
Virtual machine:	no

Run Information

Configuration:	/home/swilliam/capi/config/fprof
Date:	July 21, 2007
Version:	HP Caliper - HP-UX Itanium Version A.4.3.10
OS:	HP-UX B.11.23 U ia64
Database:	/home/swilliam/sw.db
Measurement scope:	per-process

Sampling Specification

Sampling event:	CPU_CYCLES
Sampling period:	500000 events
Sampling period variation:	25000 (5.00% of sampling period)
Sampling counter privilege:	user (user-space sampling)
Data granularity:	16 bytes
Data sampled:	IP

Run Summary for be

Target Application

Program:	/opt/langtools/bin/be
Invocation:	/opt/langtools/bin/be -keep_dir /var/tmp//00000639_...
Process ID:	27262 (exec'd by sh, Process ID 27262)
Start time:	09:07:46 AM
End time:	09:08:37 AM
Termination Status:	0
Last modified:	March 31, 2006 at 10:35 AM
Memory model:	ILP32
Processor set:	default

Target Execution Time

Real time:	51.126 seconds
User time:	50.099 seconds
System time:	0.238 seconds

Sampling Specification

Number of samples:	158261
Data sampled:	IP

Load Modules Included

Load Modules	Start Address	End Address	Full Path
--------------	---------------	-------------	-----------

GUI reports (per-process with drill down)

View per-process data

Drill down (with double-click)

The screenshot displays the HP Caliper GUI with three overlapping windows illustrating the drill-down process:

- Left Window:** Shows the 'Application Processes' list. Processes include 'be (PID: 27262)', 'ecom (PID: 27251)', 'u2comp (PID: 27253)', 'ld (PID: 27252)', 'sh (PID: 27258)', 'cc (PID: 27250)', 'sh (PID: 27259)', 'make (PID: 27253)', 'uname (PID: 27261)', 'rm (PID: 27269)', 'rm (PID: 27263)', 'uname (PID: 27260)', 'sh (PID: 27260)', 'sh (PID: 27261)', 'make (PID: 27259)', 'make (PID: 27258)', 'cc (PID: 27252)', and 'cc (PID: 27251)'. A double-click arrow points from 'be (PID: 27262)' to the next window.
- Middle Window:** Shows the 'Load Modules' for the selected process 'be'. Modules listed include 'libc.so.1', 'libcusp.so.1', 'dld.so', 'libm.so.1', and '*kernel gateway*'. A double-click arrow points from 'libc.so.1' to the next window.
- Right Window:** Shows the 'Function Disassembly' for the selected module 'libc.so.1'. The table below summarizes the data shown in this window.

Address	Op	Op	Op	IP Samples	Overview: Percent of Grand Totals
0x0000:0	M	setf.sig	f6=r33	237	0.16%
:1	M	setf.sig	f9=r32		0.32%
:2	I	adds	r16=-1,r33		
0x0010:0	M	cmp4.eq.unc	p7,p0=1,r33		
:1	M	cmp4.eq.unc	p6,p0=r0,r33		
:2	B (p7)	br.dpnt.many	{self}+0x1a...		
0x0020:0	M	and	r11=r16,r33	187	
:1	M	and	r8=r16,r32		
:2	I (p6)	break.i	1		
0x0030:0	M	adds	r15=-2,r33		
:1	M	addl	r14=-5736,r1		
:2	I	adds	r9=1,r33 ;;		
0x0040:0	M	cmp4.eq.unc	p6,p0=r0,r11	194	
:1	M	and	r11=r16,r15		
:2	I	zxt4	r15=r15		
0x0050:0	M	and	r9=r9,r33		
:1	I	zxt4	r16=r32		
:2	B (p6)	br.ret.dpnt.many	rp ;;		
0x0060:0	M	setf.sig	f11=r16	217	
:1	M	cmp4.eq.unc	p6,p0=r33,r32		
:2	I	zxt4	r8=r33		

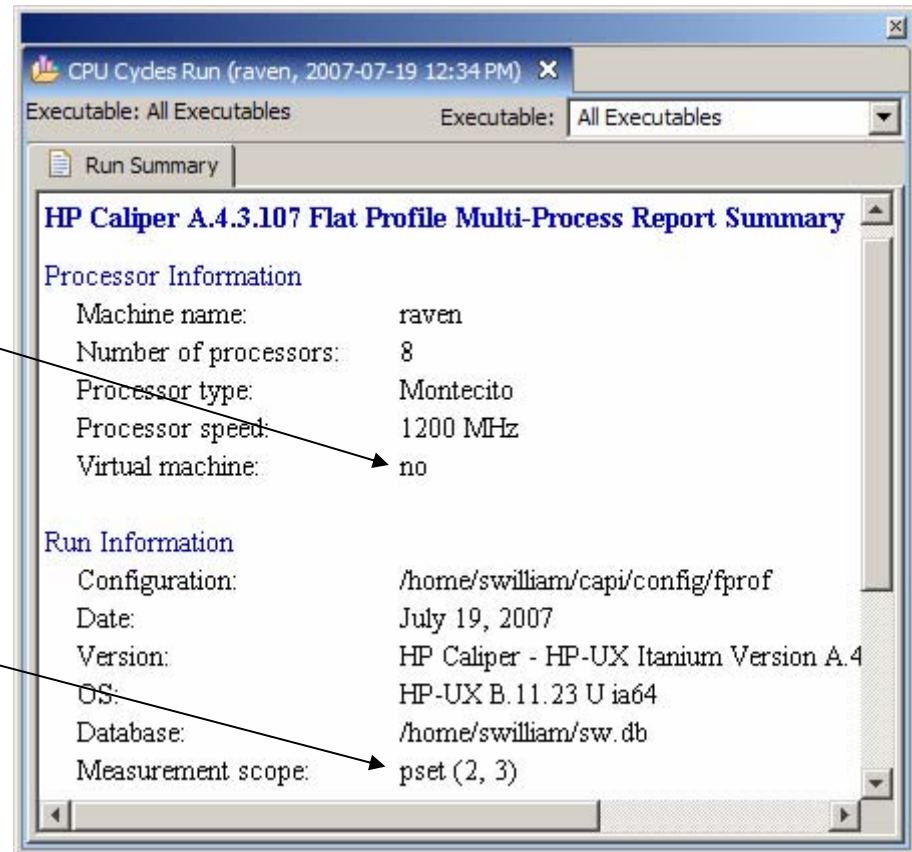
What's new in Caliper?

- VSE information in reports
- Processor set selection
- Hyper-Threading data (Dual-Core Intel® Itanium® 2 processor only)
- System usage data
- Three top-level views of data (per-process, per-load module, or per-executable)
- Profiling traps, interrupts, faults (Dual-Core Intel Itanium 2 processor only)
- Latency buckets for data cache miss samples
- Histograms of data cache misses by data address
- Call-stack sampling

VSE information in run summaries

Performance data collected on virtual machine?

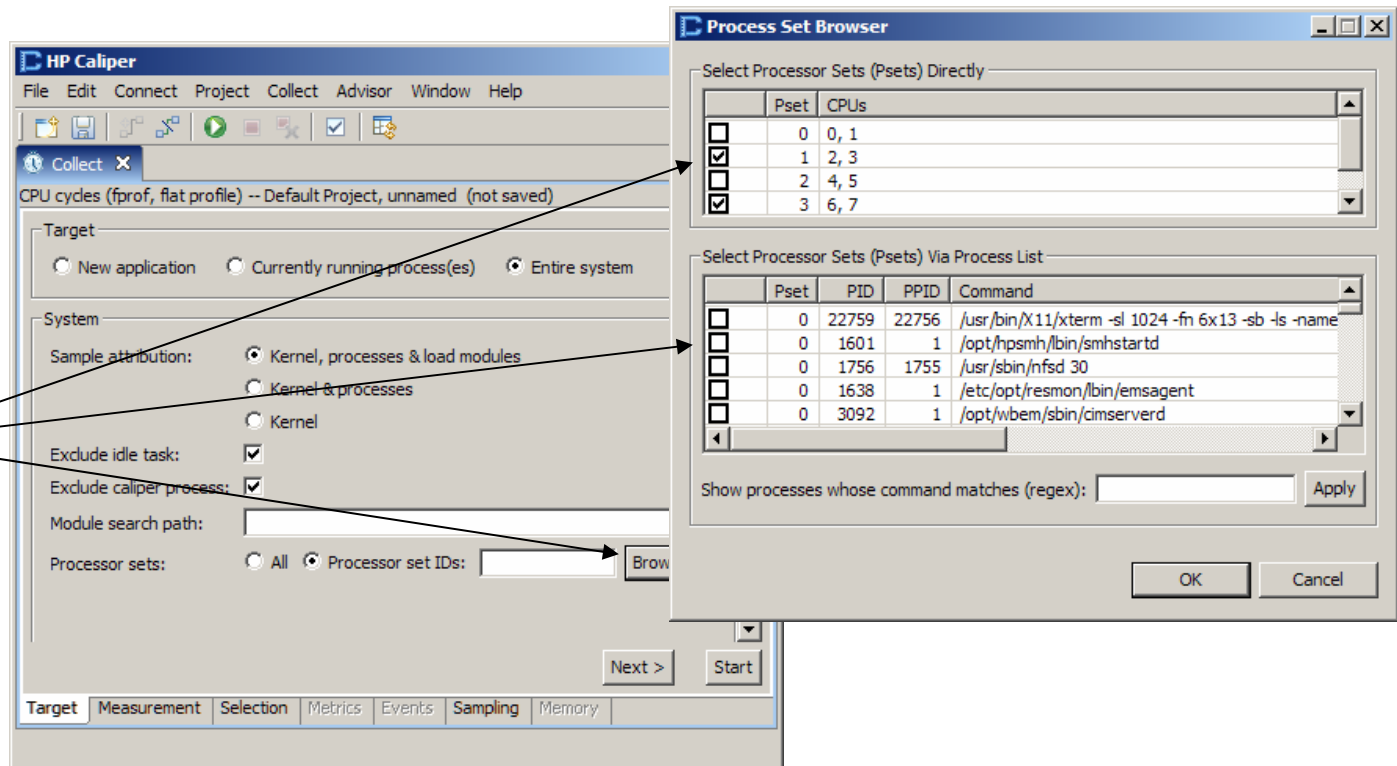
Which processor sets included in measurement?



Choosing processor sets

Command line: `--scope pset=ID[:ID...]`

GUI



Hyper-Threading

Was Hyper-Threading enabled during data collection?

HP Caliper

File Edit Connect Project Collect Advisor Window Help

Collect CPU Cycles Run (manaslu, 2007-07-20 11:05 AM) x Executable: be (1 instance)

Executable: be (1 instance)

Run Summary

Run Summary for be

Target Application

Program: /opt/langtools/sbin/be
Invocation: /opt/langtools/sbin/be -keep_dir /var/tmp//00000639_00001A20 +Udriver
Process ID: 6695 (exec'd by sh, Process ID 6695)
Start time: 11:05:24 AM
End time: 11:06:28 AM
Termination Status: 0
Last modified: October 24, 2006 at 11:33 AM
Memory model: ILP32
Processor set: 1
Hyperthreading: on for this process

Target Execution Time

Real time: 63.779 seconds
User time: 61.750 seconds
System time: 0.876 seconds

Sampling Specification

Number of samples: 171446
Data sampled: IP

Hyper-Threading: impact

Cycles used by
this process'
hyper-thread
(AT = False)

Cycles used by
both hyper-threads
(AT = True)

Percentage cycles
due to process's
hyper-thread

HP Caliper

Collect CPU Cycles Run (manaslu, 2007-07-20 11:05 AM) x

Process: be (PID: 6695)

Show Data for: Processes be (PID: 6695)

Run Summary CPU Events

Metrics Summed for be (PID: 6695)

Event Name	PLM U.K	Threshold	AC	AT	Count
BE_L1D_FPU_BUBBLE.ALL	x__	0	T	F	4663185979
BE_RSE_BUBBLE.ALL	x__	0	F	F	2155584005
BE_FLUSH_BUBBLE.ALL	x__	0	T	F	5246162626
BACK_END_BUBBLE.FE	x__	0	F	F	4392875287
CPU_OP_CYCLES.ALL	x__	0	T	F	85709723329
CPU_OP_CYCLES.ALL	x__	0	T	T	86354119100
BE_FVE_BUBBLE.ALL	x__	0	F	F	15040210070

% Cycles lost due to sta
5.44 = 100 * (BE_L1D_FPU_BUBBLE.ALL - BE_L1D_FPU_BUBBLE.L1D) / CPU

% Cycles lost due to reg
0.63 = (100 * (BE_FLUSH_BUBBLE.ALL - BE_FLUSH_BUBBLE.L1D) / CPU)

% Cycles lost due to GF
0.35 = 100 * (BE_RSE_BUBBLE.ALL - BE_RSE_BUBBLE.L1D) / CPU

% Cycles lost due to FPU (floating-point unit) stalls
0.00 = 100 * (BE_L1D_FPU_BUBBLE.ALL - BE_L1D_FPU_BUBBLE.L1D) / CPU

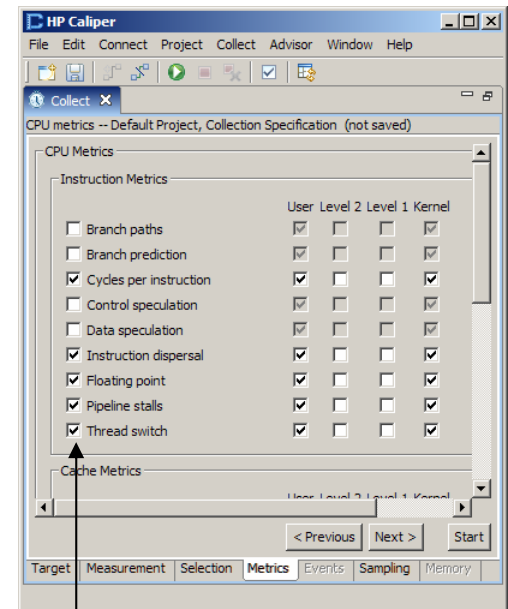
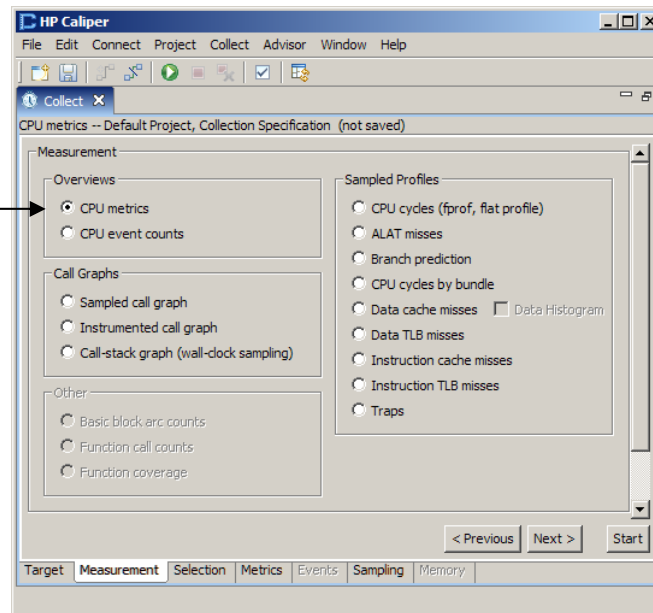
% Core cycles due to this thread
99.25 = 100 * (CPU_OP_CYCLES.ALL / CPU_OP_CYCLES.ALL:at=true)

Note: available with alat, cycles, ecount, and fprof measurements

Hyper-Threading: collecting thread switching data

Command line: `caliper cpu --metric overview ...`
`caliper cpu --metric threadswitch ...`

GUI



Thread switch data collected by default

Hyper-Threading: thread switch (TS) data

Distribution of stall cycles before TSs

Breakdown of TS causes

Total TS overhead cycles per second

TSs per 1000 instructions

Total TS gated cycles per second

TSs per second

Entire run

Header tool tip

Sample points

The screenshot shows the HP Caliper interface with two tables. The top table is titled 'THREADSWITCH Summary Statistics' and the bottom table is 'THREADSWITCH Metric Samples'. Both tables have columns for 'Cause of TS', 'Stall Cycles Before TS', 'Overhead Cycles Per Sec', and 'Gated Cycles Per Sec'. A yellow tooltip points to the 'Overhead Cycles Per Sec' column in the summary table, displaying the text 'Total Thread Switch Overhead Cycles per Second'.

Stats	TS Per Sec	TS Per Kinst	Cause of TS				Stall Cycles Before TS					Overhead Cycles Per Sec	Gated Cycles Per Sec
			L3miss	Timer	Hint	Other	0-3	4-15	16-63	64-255	>=256		
MEAN	371934	0.167	61.56%	37.32%	0.00%	1.12%	50.26%	45.81%	3.77%	0.12%	0.04%	775380	10710
STDEV	245631	0.170	15.33%	14.85%	0.00%	0.71%	12.41%	13.28%	1.92%	0.06%	0.03%	499013	18092
90%CI	64696	0.045	4.04%	3.91%	0.00%	0.19%	3.27%	3.50%	0.51%	0.02%	0.01%	240208	2175
MINIMUM	115956	0.054	23.13%	6.27%	0.00%	0.23%	13.40%	27.59%	0.70%	0.02%	0.00%	643947	5945
LOW90	307238	0.122	57.52%	33.41%	0.00%	0.93%	46.99%	42.31%	3.27%	0.10%	0.03%	906813	15475
HIGH90	436630	0.211	65.60%	41.23%	0.00%	1.31%	53.53%	49.31%	4.28%	0.14%	0.05%	2818994	119738
MAXIMUM	1396500	1.047	93.36%	74.79%	0.00%	3.51%	69.31%	84.12%	8.31%	0.28%	0.11%		

Sample	TS Per Sec	TS Per Kinst	Cause of TS				Stall Cycles Before TS					Overhead Cycles Per Sec	Gated Cycles Per Sec
			L3miss	Timer	Hint	Other	0-3	4-15	16-63	64-255	>=256		
1	401413	0.170	49.79%	49.73%	0.00%	0.48%	63.77%	32.24%	3.66%	0.22%	0.11%	1176963	119738
2	245413	0.092	58.13%	41.26%	0.00%	0.61%	55.81%	39.31%	4.73%	0.13%	0.02%	511450	12544
3	249081	0.089	57.23%	41.88%	0.00%	0.89%	60.98%	34.53%	4.31%	0.16%	0.02%	537706	11263
4	221694	0.079	55.46%	43.48%	0.00%	1.06%	56.12%	39.60%	4.08%	0.14%	0.06%	467713	10356
5	271169	0.097	64.48%	34.47%	0.00%	1.06%	48.40%	48.41%	3.08%	0.09%	0.03%	554388	6881

System usage data

- Currently available only from command line tool and text reports
- Collected using kernel instrumentation
- Collected when `-system-usage` option is specified:
 - `--system-usage [all][:runstatus][:syscalls]`
- `runstatus` - Results in a report on how much time each process spent running, eligible to run but not running, and waiting
- `syscalls` - Results in a report of how much time a process spent in each system call as well as how many times each system call was called
- `all` -- Equivalent to specifying both `runstatus` and `syscalls`

System usage data (sample report)

System Usage - Run Status Breakdown of process's time*

```

-----
Relative      ----- Time (thread secs) -----      ----- Percentage -----
Time          Running      Eligible      Waiting      Running      Eligible      Waiting
-----
Overall      22.6300      0.0030      79.8851      22.07%      0.00%      77.92%
-----

```

System Usage - Syscalls Breakdown of process's system calls

```

-----
Relative Time          Total      Calls ----- Time (thread secs) -----
System Call          Calls      / Sec      Minimum      Average      Maximum      Total
-----
Overall
open                 1044      20.47      0.00001      0.00014      0.00546      0.14749
execve                1         0.02      0.12050      0.12050      0.12050      0.12050
brk                   40636     796.78     0.00000      0.00000      0.00152      0.04566
write                 641       12.57     0.00000      0.00004      0.00869      0.02733
read                  49         0.96     0.00000      0.00025      0.00368      0.01206
unlink                1          0.02     0.00501      0.00501      0.00501      0.00501

```

*Time is summed across threads (can be > process time)

Three top-level data views

- Top-level profile data can be shown in three separate ways:
 - By process
 - By load module (data combined for each load module instance across all processes measured)
 - By executable (data combined for process instances sharing same executable)
- Lower-level data (e.g., per-function data) reflects top-level data (e.g., load module's function data for all processes combined)

Top-level views (text reports)

Process Summary `--group-by none`

% Total IP Samples	Cumulat % of Total	IP Samples	Process
27.24	27.24	43821	be (PID: 26928)
12.97	40.20	20858	be (PID: 26897)
7.67	47.87	12339	be (PID: 26910)
6.77	54.64	10889	be (PID: 26887)
4.19	58.83	6736	be (PID: 26881)
4.18	63.01	6731	be (PID: 26914)
3.74	66.76	6023	be (PID: 26974)
2.95	69.71	4747	ecom (PID: 26834)
2.76	72.47	4447	be (PID: 26980)
2.66	75.13	4273	be (PID: 26905)

[Minimum process entries: 5, percent cutoff: 2.00,
cumulative percent cutoff: 100.00]

`--group-by executable`

Process Summary

% Total IP Samples	Cumulat % of Total	IP Samples	Process
87.13	87.13	140169	be (38 instances)
10.84	97.97	17444	ecom (38 instances)
1.60	99.57	2569	u2comp (PID: 26859)
0.24	99.81	382	sh (78 instances)
0.11	99.91	170	ld (PID: 26857)

[Minimum process entries: 5, percent cutoff: 2.00,
cumulative percent cutoff: 100.00]

`--group-by module`

Load Module Summary

% Total IP Samples	Cumulat % of Total	IP Samples	Load Module
71.15	71.15	114468	be
16.75	87.91	26953	libc.so.1
7.92	95.83	12749	ecom
1.96	97.79	3152	libCsup.so.1
1.14	98.94	1841	u2comp
0.78	99.71	1249	dld.so
0.14	99.85	227	sh
0.08	99.94	134	ld
0.03	99.97	48	make
0.02	99.99	34	*kernel gateway*
0.00	99.99	7	libstream.so.1
0.00	100.00	6	cc
0.00	100.00	4	libm.so.1
0.00	100.00	2	libunwind.so.1
0.00	100.00	1	libstd.so.1
0.00	100.00	1	libdl.so.1
100.00	100.00	160876	Total

Top-level views (GUI)

Choose top level

The image displays three overlapping screenshots of the HP Caliper GUI, illustrating different top-level views. The text "Choose top level" is positioned above the screenshots, with arrows pointing to the "Top Level" dropdown menus in each view.

Processes View: The "Top Level" is set to "Processes". The table shows IP Samples for various processes, with "be (PID: 26928)" having the highest value at 43821.

Process	IP Samples
be (PID: 26928)	43821
be (PID: 26897)	20858
be (PID: 26910)	12339
be (PID: 26887)	10889
be (PID: 26881)	6736
be (PID: 26914)	6731
be (PID: 26974)	6023
ecom (PID: 26834)	4747
be (PID: 26980)	4447
be (PID: 26905)	4273
u2comp (PID: 26859)	2569
be (PID: 26901)	2499
be (PID: 26929)	2406
ecom (PID: 26823)	2290
be (PID: 26898)	2125
be (PID: 26911)	2049
be (PID: 26956)	1901
ecom (PID: 26821)	1867
be (PID: 26959)	1462
be (PID: 26953)	1366
be (PID: 26971)	1217

Executables View: The "Top Level" is set to "Executables". The table shows IP Samples for various executables, with "be (38 instances)" having the highest value at 140169.

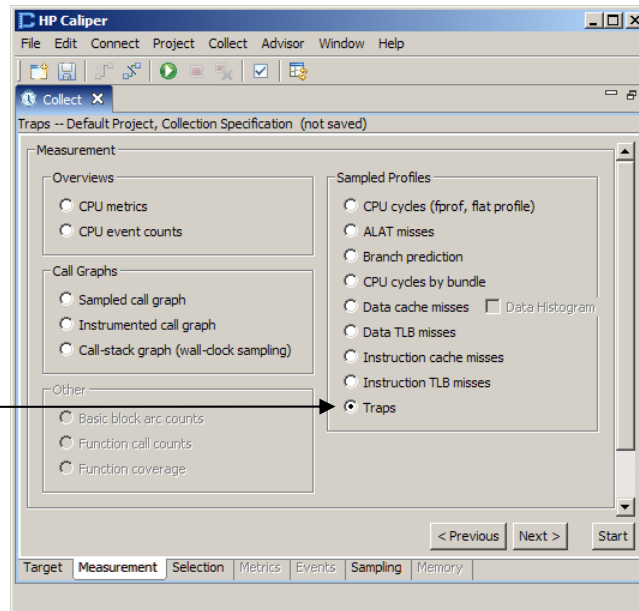
Executable	IP Samples
be (38 instances)	140169
ecom (38 instances)	17444
u2comp (1 instance)	2569
sh (78 instances)	382
ld (1 instance)	170
make (40 instances)	75
uname (39 instances)	46
cc (40 instances)	17
rm (2 instances)	4

Load Modules View: The "Top Level" is set to "Load Modules". The table shows IP Samples for various load modules, with "be" having the highest value at 114468.

Load Module	IP Samples
be	114468
libc.so.1	26953
ecom	12749
libc.so.1	3152
u2comp	1841
dld.so	1249
sh	227
ld	134
make	48
kernel gateway	34
libstream.so.1	7
cc	6
libm.so.1	4
libunwind.so.1	2
libstd.so.1	1
libdl.so.1	1

Profiling traps, faults, and interrupts

- Traps captured by sampling Montecito ETB
- All 34 traps, interrupts, and faults sampled
- Drill down into trap samples, down to instruction level
- Command line: `caliper traps {-w | PIDs | program ...}`
- GUI



Viewing traps, interrupts, faults profile

Command line: `--traps-reported trap1[,trap2,...,trap6]`

Show traps by load module

Choose traps of interest

The screenshot shows the HP Caliper interface with a 'Traps Run' profile. The main window displays a histogram of traps by load module. The 'Trap Samples' column is expanded to show a table of trap counts for various modules. The 'Choose Traps' dialog box is open, allowing the user to select specific traps to display in the histogram.

Trap Samples	ITLB	DTLB	GEYCP	UADREF	FPFLT	FPTRP	OTHER
be	1499	0	1488	0	0	1	0
ecom	37	0	35	0	1	0	1
libc.so.1	13	0	1	0	0	0	12
u2comp	8	0	4	0	0	0	4
dld.so	2	0	2	0	0	0	0
vmunix	1	0	0	0	0	0	1

Choose Traps
Choose traps to be displayed (maximum of 6)

- Instruction TLB Fault
- Data TLB Fault
- Alternate Instruction TLB Fault
- Alternate Data TLB Fault
- Data Nested TLB Fault
- Instruction Key Miss Fault
- Data Key Miss Fault
- Data Dirty Bit Fault
- Instruction Access Bit Fault
- Data Access Bit Fault
- Break Instruction Fault
- External Interrupt
- Virtual External Interrupt
- Page Not Present Fault
- Kernel Permission Fault
- Instruction Access Rights Fault
- Data Access Rights Fault
- General Exception Fault: Unimplemented Data Address, Illegal Oper
- Disabled FP Register Fault
- NAT Consumption Fault
- Speculative Operation Fault

Note: Only the top-level histogram tab will remain open if you select OK.

Trap abbreviations for command line

```
$ caliper traps --traps-reported foo vand
```

 Use bad trap name ('foo') to get list

```
HP Caliper: usage error:
```

```
Unrecognized trap name ("foo") specified for "--traps-reported".
```

```
Specify a comma-separated list of up to six of the following trap types:
```

```
32incpt, 32inrpt, adtlb, aitlb, break, daccs, darght, dbit, debug, dfpreg  
dkey, dntlb, dtlb, fpflt, fptrp, gexcp, ia32exp, iaccs, iarght, ikey  
int, itlb, kperm, lptrp, natc, pnotp, specop, sstrp, tbtrp, uadref  
usdref, vfault, vhpt, vint
```

```
For a descriptive list of trap types, run:
```

```
caliper info -r traps
```

```
$ caliper info -r traps
```

 Use info argument to get detailed list

```
=====  
HP Caliper A.4.3.107 Report Information  
=====
```

```
.  
. .  
32INCPT  
  
        IA32 Intercept  
  
32INRPT  
  
        IA32 Interrupt  
  
. .  
. .  
. .
```

Bucketing data cache miss latencies

- Average and total latency is not sufficient
 - Memory system is more complex on cell-based systems
 - Data distribution/affinity has big impact on performance
- Need detailed breakdown of access latencies across the memory hierarchy
- Each sampled latency recorded in one of eight buckets
 - L2, L3, local c2c, cell local memory, etc.
 - Bucketed latencies available down to instruction level
- Uses expected system latency
 - Based on chipset type, CPU type, and frequency

Viewing latency buckets

Cache-to-cache exchange between two CPUs (same FSB*)

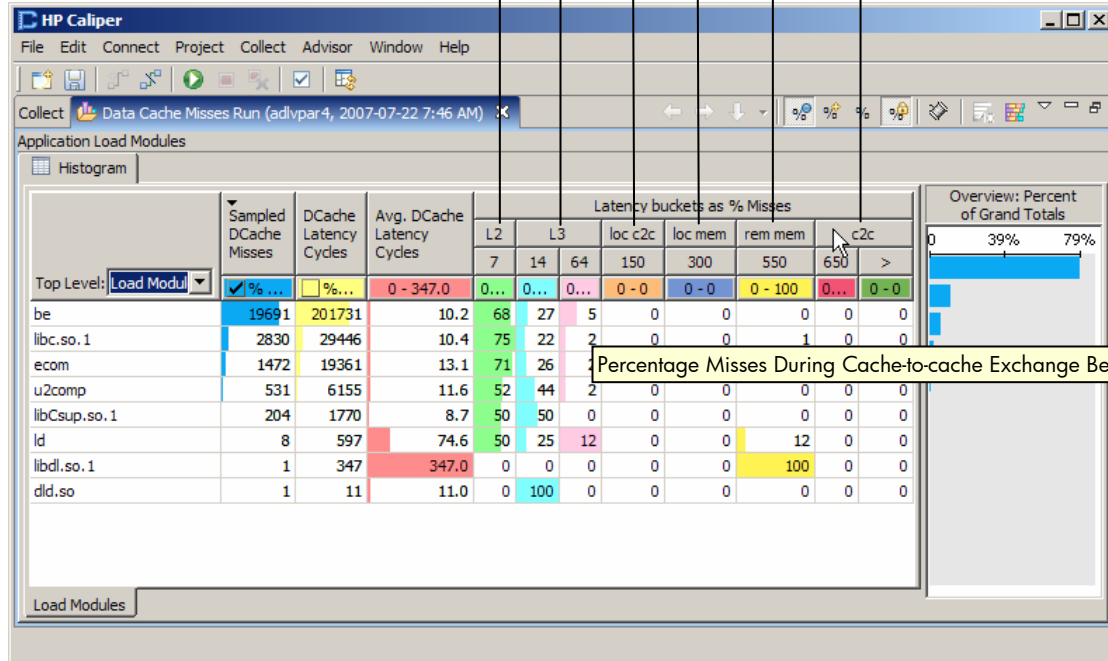
Memory in local cell

Memory in remote cell

Cache-to-cache exchange between two CPUs (different FSBs*)

L3 cache

L2 cache



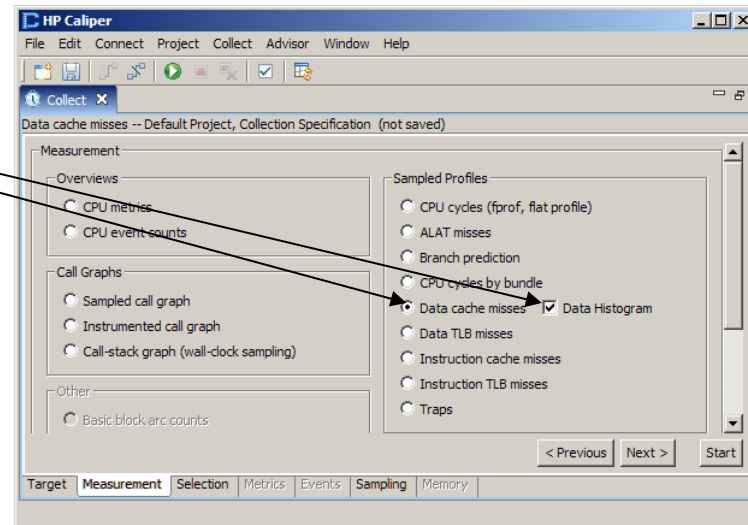
Percentage Misses During Cache-to-cache Exchange Between Two CPUs in Different FSBs (Front Side Buses)

c2c
tool tip

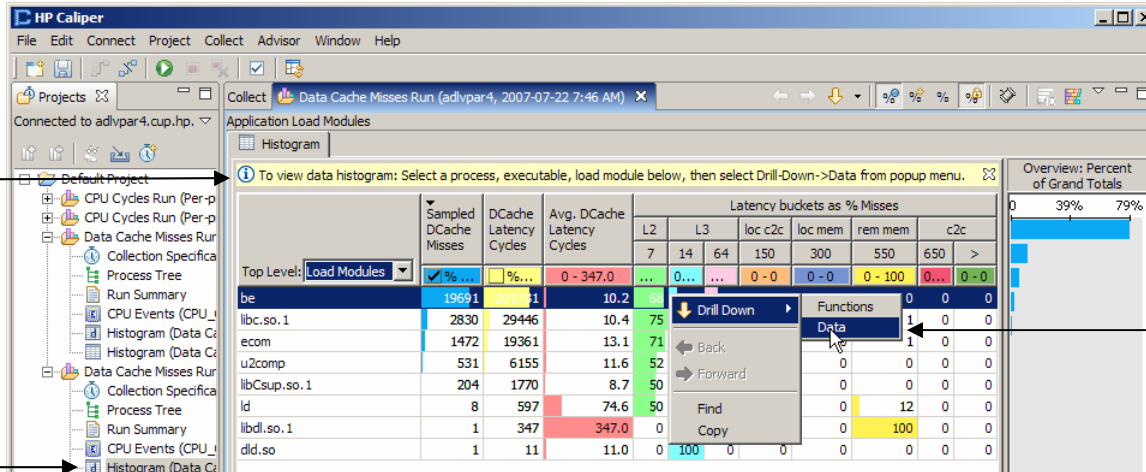
*FSB = front side bus

Data cache misses by data address

- Add-on for data cache misses (dcache) measurement
- Collects histogram of data addresses that were accessed when sampled data cache misses occurred
- Data addresses mapped to global variables or process regions
- Command line: `caliper dcache --data-summary ...`
- GUI



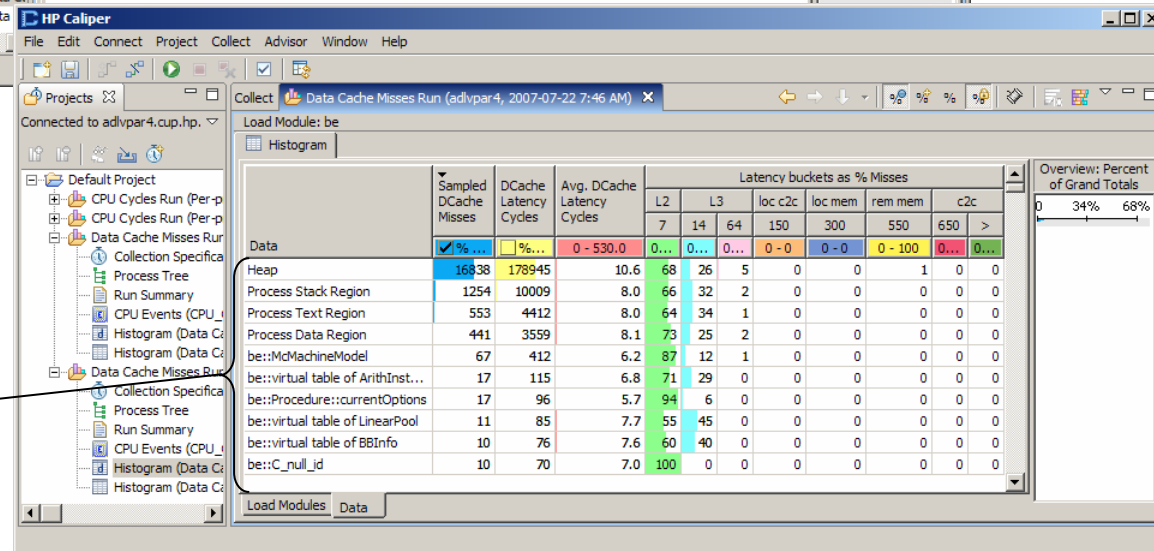
Viewing data cache misses by address



Drill down to data histogram for be

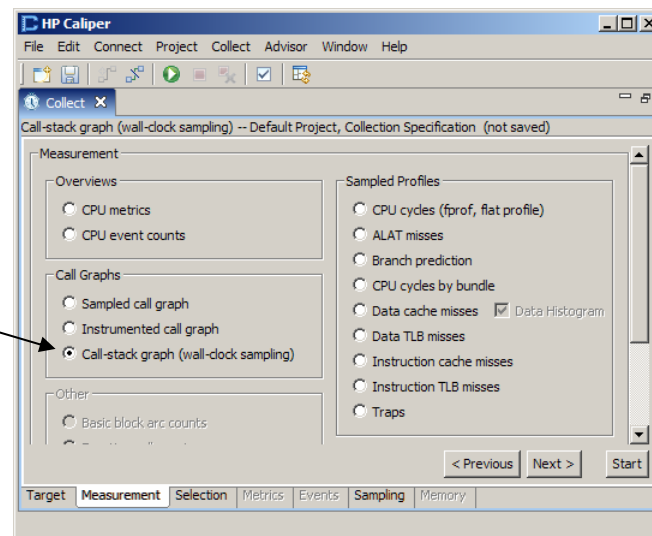
Guidance provided if data histogram chosen

Process regions and global variables



Call stack sampling

- Real-time sampling of call stack
- Helps locate where the program is waiting for system calls, locks, or I/O
- Command line: `caliper cstack ...`
- `--frame-depth` sets maximum frame depth recorded (default: 32)
- GUI



cstack text report (hot call paths)

Hot Call Paths

```
-----  
--% Total Hits In Only--  
Run +      Run      Block  
Block      Hits      Hits      Name  
Hits      Only      Only  
-----  
  1.1      1.1      0.0 be::DagNode::dagConstMarkPredArc(DagNode*,DagNode*,Dag*) [138]  
      be::Dag::addNonRegisterDependences(DagNode*,unsigned int,DagAttributes&,unsigned int&,unsigned int&) [77]  
      be::Dag::addInstruction(Instruction*,unsigned int) [51]  
      be::Dag::addInstsInBB(Instruction*) [47]  
      be::Dag::Dag(RegionNode&, ...,PostDominatorRelation*)(shared) [49]  
      be::Dag::Dag(RegionNode&,SchedulerPhase,ComponentOptions*, ...,PostDominatorRelation*)(complete) [50]  
      be::RegionNode::schedule(ComponentOptions*,PredQuerySystem*,unsigned int) [19]  
      be::Procedure::schedule() [15]  
      be::Procedure::optimize(Tahoe_asm_section*) [11]  
      be::TG_ProcedureHandle::Optimize(Tahoe_asm_section*) [12]  
      be::TG_TailGen::Perform_() [9]  
      be::TG_TailGen::TG_TailGen(SYZ_RoutineScope*,bool,bool)(shared) [8]  
      be::TG_PerformTailGeneration(SYZ_RoutineScope*) [10]  
      be::SYZ_Driver::RoutineScopeBEOpt_(SYZ_RoutineScope*,bool,bool,bool) [3]  
      be::SYZ_Driver::PerformBEOptimizations_(SYZ_LinkUnit*,bool,bool) [4]  
      be::SYZ_Driver::BEStart_(char**) [2]  
      be::SYZ_Driver::StandaloneBE(int,char**) [5]  
      be::main [7]  
      dld.so::main_opd_entry [6]  
-----  
  0.9      0.9      0.0 be::ResourceManager::hasReverseSiblingResources(ResourceId) const [76]  
      be::BBInfo::addLastDef(ResourceId,LastDef*) [67]  
      be::LocalOptimizer::updateLastDefTable(Instruction*,ValueNumber) [61]  
      be::LocalOptimizer::summarizeInst(Instruction*,ValueNumber) [57]  
      be::LocalOptimizer::forwardLocalOpt(BasicBlock&,BBInfo*) [37]  
      be::LocalOptimizer::localOptExtendedBasicBlock(BasicBlock&,BBInfo*,JoinPointInfo*) [38]  
      be::LocalOptimizer::localOptExtendedBasicBlock(BasicBlock&,BBInfo*,JoinPointInfo*) [38]  
      be::LocalOptimizer::localOpt(BasicBlock*) [45]  
      be::Procedure::localOpt(ComponentOptions*) [41]  
      be::Procedure::schedule() [15]  
      .  
      .  
      .
```

cstack text report (call stack graph)

Call Graph

```

-----
--% Total Hits In / Under--  % Func Hits
Run +      Run      Block      Under Parent      Parents
Index Block  Hits      Hits      In Func      Name      Index
Hits      Only      Only      In Children      Name      Children
-----
[1]  100.0   100.0    0.0      0.00          *ROOT* [1]
      95.61    dld.so::main_opd_entry [6]
      0.88     be::OPT_FUDBuilder::Perform_(SYZ_BBNode*) [188]
      0.53     dld.so::EM_mark_BOS [283]
      0.35     be::SYZ_Driver::PerformBEOptimizations_(SYZ_LinkUnit*,bool,bool) [4]
      0.18     be::LNO_LoopOptimization::LNO_LoopOptimization(SYZ_RoutineScope*)(shared) [21]
      0.18     be::TG_ProcedureHandle::Optimize(Tahoe_asm_section*) [12]
      0.18     uld.so::$START$ [815]
      0.18     be::OPT_SSAPRE::ProcessIncomingDefsForPhi_(SYZ_BBNode*, *,allocator>*) [148]
-----
      99.82    be::SYZ_Driver::StandaloneBE(int,char**) [5]
      0.18    *ROOT* [1]
[2]  95.8     95.8     0.0      0.00          be::SYZ_Driver::BEStart_(char**) [2]
      99.63    be::SYZ_Driver::PerformBEOptimizations_(SYZ_LinkUnit*,bool,bool) [4]
      0.37    be::$cold__ZN10SYZ_Driver23PerformBEOptimizations_EP12SYZ_LinkUnitbb [423]
-----
      99.82    be::SYZ_Driver::PerformBEOptimizations_(SYZ_LinkUnit*,bool,bool) [4]
      0.18    *ROOT* [1]
[3]  95.8     95.8     0.0      0.00          be::SYZ_Driver::RoutineScopeBEOpt_(SYZ_RoutineScope*,bool,bool,bool) [3]
      44.04    be::TG_PerformTailGeneration(SYZ_RoutineScope*) [10]
      43.12    be::OPT_PerformScalarOptimization(SYZ_RoutineScope*,bool,bool) [13]
      12.29    be::LNO_PerformLoopOptimization(SYZ_RoutineScope*) [24]
      0.37    be::SYZ_PerformBitAccessLowering(SYZ_RoutineScope*) [350]
      0.18    be::OPT_ComputeAliasRelationship(SYZ_RoutineScope*,bool) [207]
-----
      99.63    be::SYZ_Driver::BEStart_(char**) [2]
      0.37    *ROOT* [1]
[4]  95.8     95.8     0.0      0.00          be::SYZ_Driver::PerformBEOptimizations_(SYZ_LinkUnit*,bool,bool) [4]
      .
      .
      .

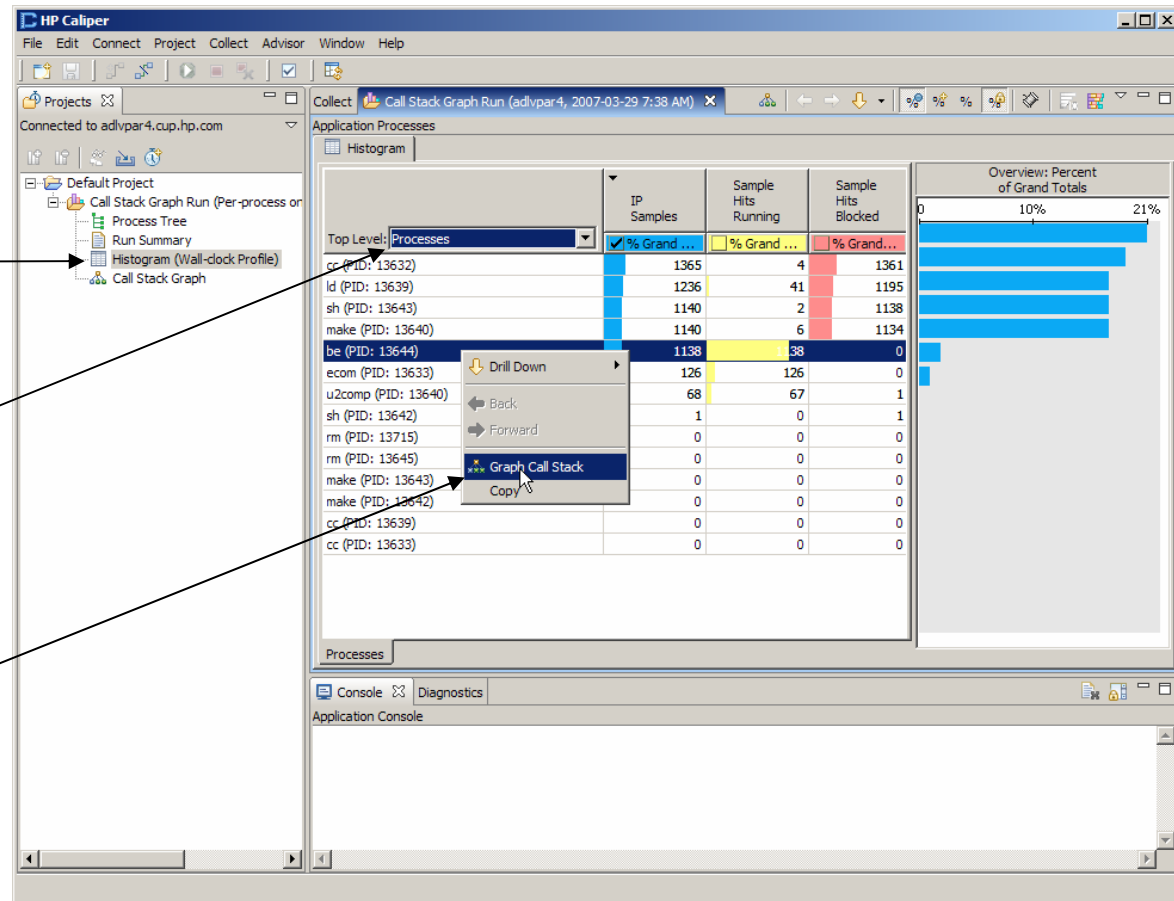
```

Viewing call stack data for a process

View call stack histogram

Choose 'Processes'

Graph process be's call stack samples



Graphing hot call stacks

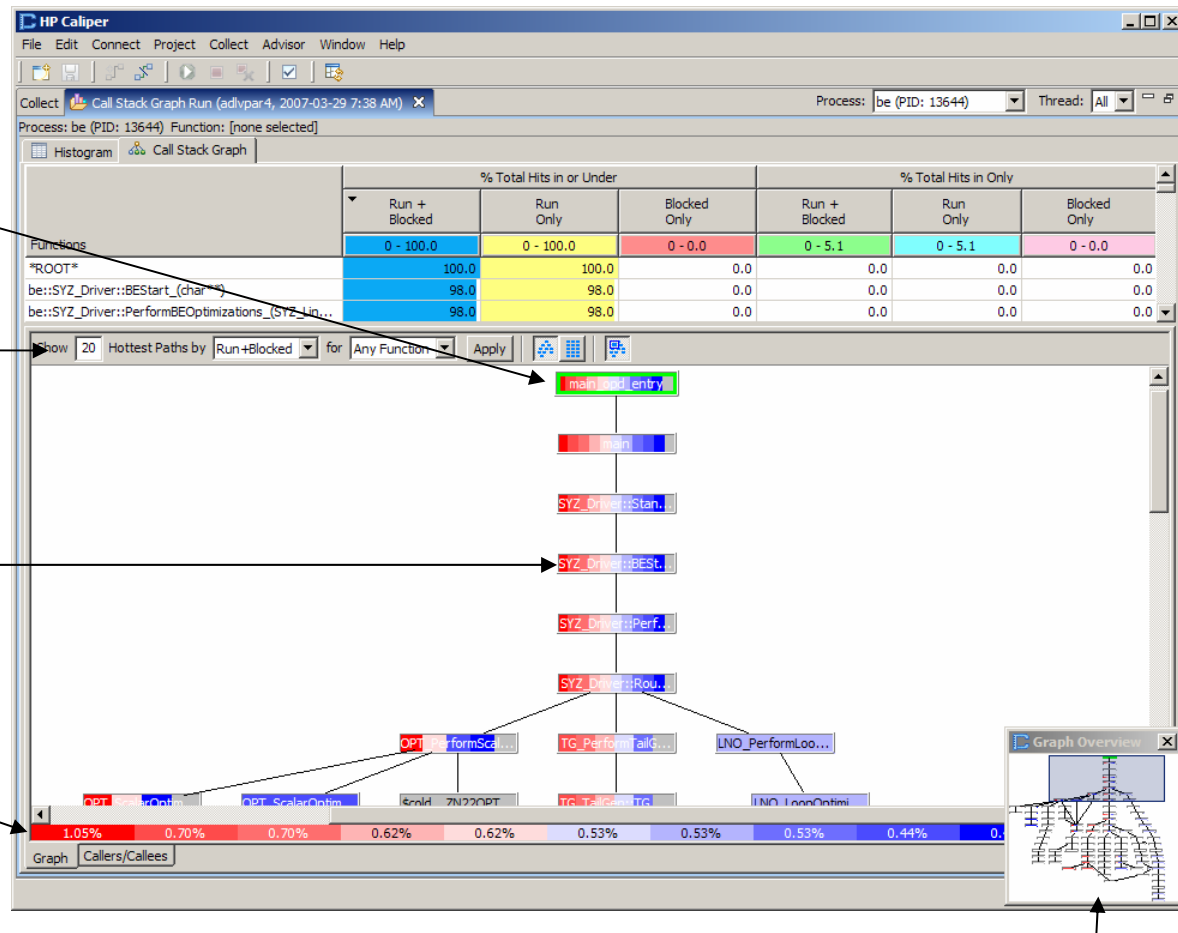
Roots highlighted in green

Maximum paths

Node colored for each path it's a member of

Legend

Overview and scroller



Note: Run/Block breakdown is not available on Linux (but can be inferred).

Viewing hot call stacks separately

Show paths separately

Double-click node to see entry in table

Functions	% Total Hits in or Under			% Total Hits in Only		
	Run + Blocked	Run Only	Blocked Only	Run + Blocked	Run Only	Blocked Only
dld.so::main_opd_entry	0 - 100.0	0 - 100.0	0 - 0.0	0 - 5.1	0 - 5.1	0 - 0.0
be::TG_PerformTailGeneration(SYZ_RoutineScope*)	97.5	97.5	0.0	0.0	0.0	0.0
be::TG_TailGen::Perform_0	43.2	43.2	0.0	0.0	0.0	0.0

Graph Callers/Callees

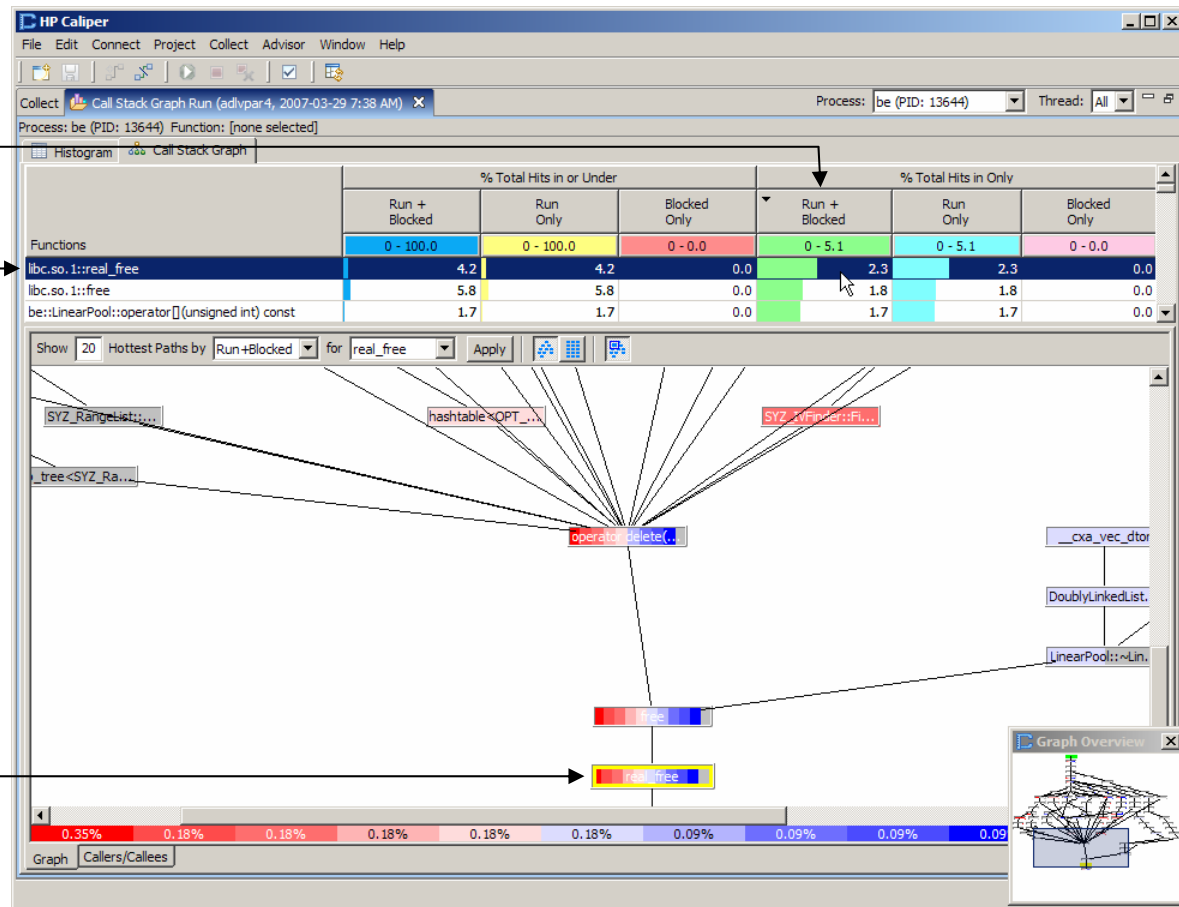
Graph Overview

Graphing a particular hot function

Sort on "Hits in Only"

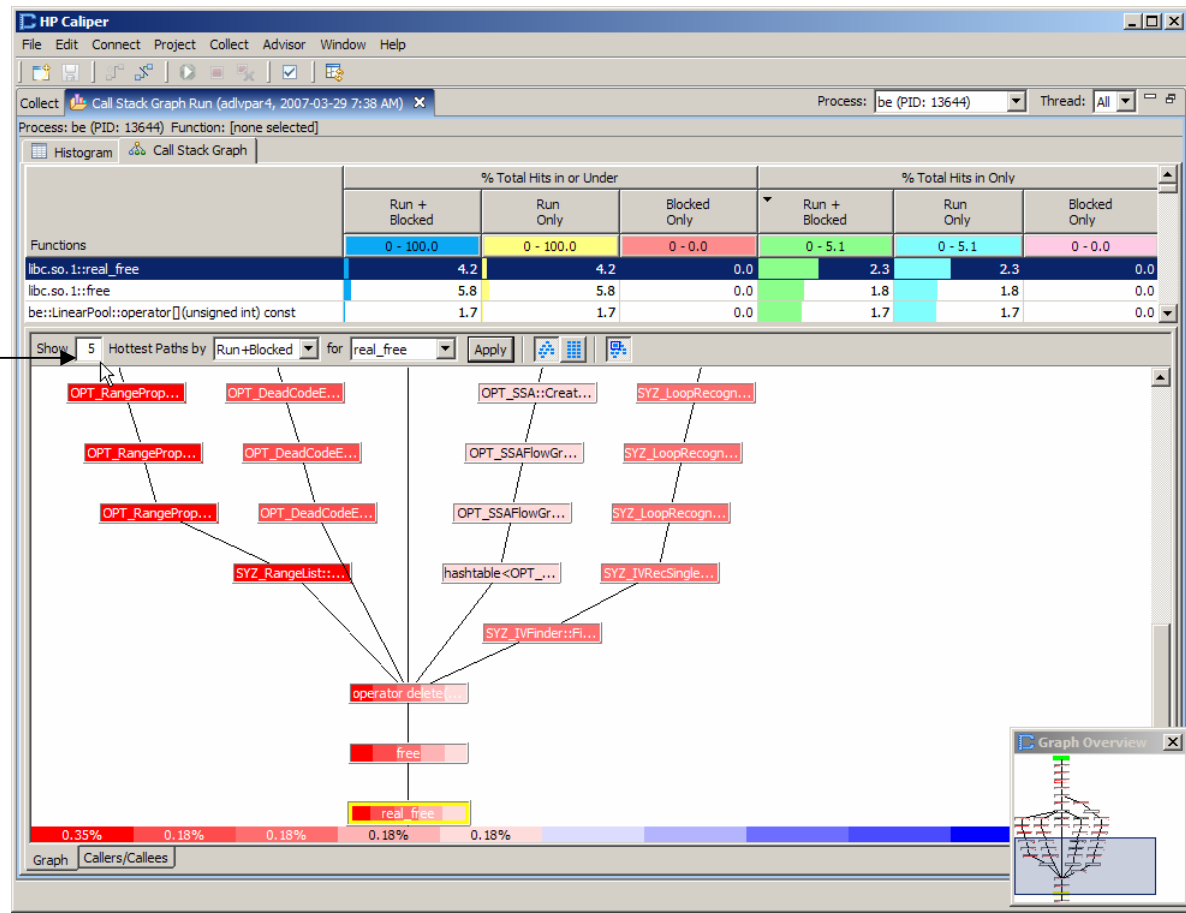
Double-click hot function

Graphed function highlighted in yellow



Simplifying the graph

Show five hottest paths that include selected function

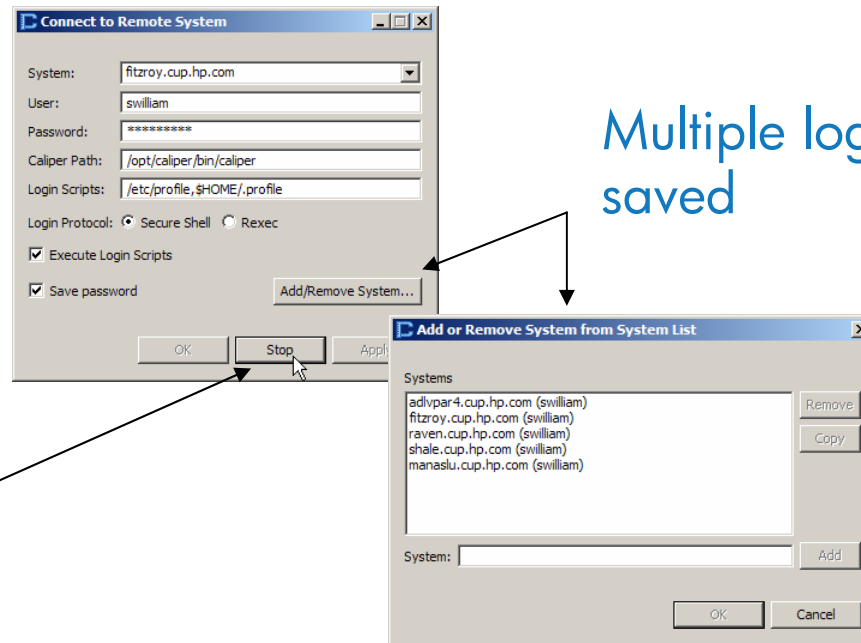


New GUI features

- Login improvements
- Call stack viewer extended to cgprof and scgprof
- Merging and differencing support
- Caliper Advisor

Login improvements

Interruptible
login

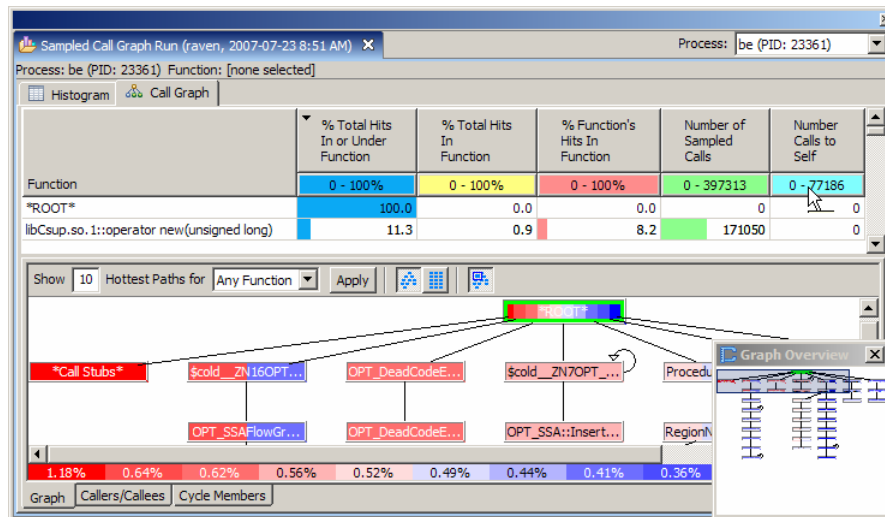


Multiple logins
saved

All communication channels forwarded through port 22 (SSH only)

Call stack viewer extended

- cgprof and scgprof measurements can now be viewed graphically.
- Viewer graphs hottest paths instead of hottest call stacks
- Paths graphed may or may not be actual call paths.
- Advantage: scgprof is lightweight



Merging and differencing in GUI

The screenshot shows the HP Caliper GUI with a context menu open over the 'Data Cache Misses Run' item. The menu includes options like 'Merge Runs', 'Diff Runs', 'Copy', and 'Paste'. The main window displays a project tree on the left and a data table on the right. The table is titled 'Merged Runs (2007-07-23 9:21 AM)' and shows various executables with their performance metrics.

Top Level: Executables	Sampled DCache Misses	DCache Latency Cycles	Avg. DCache Latency Cycles	Latency buckets as % Misses							IP Samples
				L2		L3			Memory		
				7	14	64	150	250	350	450	
be (2 instances)	20237	189712	9.4	67	27	5	0	0	0	0	158261
ecom (2 instances)	1386	12698	9.2	65	30	5	0	0	0	0	15385
u2comp (2 instances)	249	2195	8.8	59	35	6	0	0	0	0	2247
ld (2 instances)	9	192	21.3	78	11	0	11	0	0	0	128
sh (10 instances)	1	5	5.0	100	0	0	0	0	0	0	19
cc (6 instances)	0	0	0	0	0	0	0	0	0	0	9
rm (4 instances)	0	0	0	0	0	0	0	0	0	0	3
uname (4 instances)	0	0	0	0	0	0	0	0	0	0	3
make (6 instances)	0	0	0	0	0	0	0	0	0	0	9
pwd (2 instances)	0	0	0	0	0	0	0	0	0	0	0

Merge fprof and dcache measurements

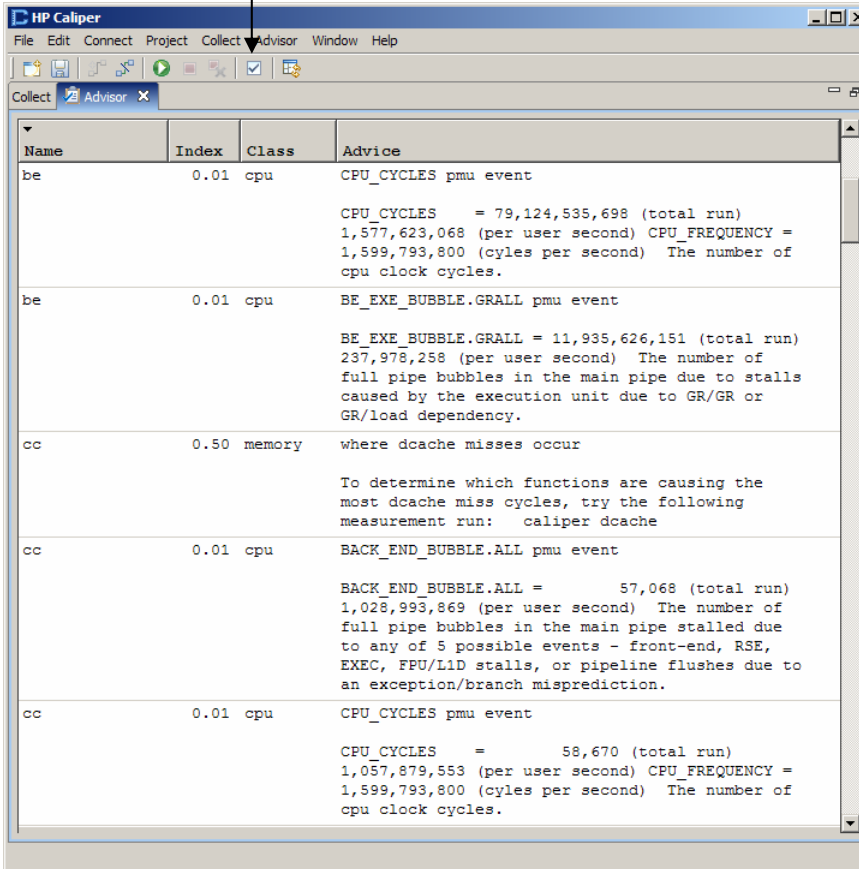
View merged data

dcache data

fprof data

Caliper Advisor in GUI

Generate advice



The screenshot shows the HP Caliper Advisor GUI. The window title is "HP Caliper" and the menu bar includes "File", "Edit", "Connect", "Project", "Collect", "Advisor", "Window", and "Help". The "Advisor" tab is active, displaying a table of performance advice. The table has four columns: "Name", "Index", "Class", and "Advice".

Name	Index	Class	Advice
be	0.01	cpu	<p>CPU_CYCLES pmu event</p> <p>CPU_CYCLES = 79,124,535,698 (total run) 1,577,623,068 (per user second) CPU_FREQUENCY = 1,599,793,800 (cycles per second) The number of cpu clock cycles.</p>
be	0.01	cpu	<p>BE_EXE_BUBBLE.GRALL pmu event</p> <p>BE_EXE_BUBBLE.GRALL = 11,935,626,151 (total run) 237,978,258 (per user second) The number of full pipe bubbles in the main pipe due to stalls caused by the execution unit due to GR/GR or GR/load dependency.</p>
cc	0.50	memory	<p>where dcache misses occur</p> <p>To determine which functions are causing the most dcache miss cycles, try the following measurement run: caliper dcache</p>
cc	0.01	cpu	<p>BACK_END_BUBBLE.ALL pmu event</p> <p>BACK_END_BUBBLE.ALL = 57,068 (total run) 1,028,993,869 (per user second) The number of full pipe bubbles in the main pipe stalled due to any of 5 possible events - front-end, RSE, EXEC, FPU/L1D stalls, or pipeline flushes due to an exception/branch misprediction.</p>
cc	0.01	cpu	<p>CPU_CYCLES pmu event</p> <p>CPU_CYCLES = 58,670 (total run) 1,057,879,553 (per user second) CPU_FREQUENCY = 1,599,793,800 (cycles per second) The number of cpu clock cycles.</p>

Future directions

- Offline viewing of performance data
- cstack in kernel space
- Data cache misses by data type
- Additional KI data consumption and reporting
- Improved disassembly viewer

Summary

- HP Caliper is a multi-OS, Integrity-server-specific performance measurement and analysis tool:
 - The supported application performance tool on HP-UX
 - HP value-add to Linux
- This “Swiss Army knife” is not a single-measurement tool:
 - Many different measurements
 - Common user interface and options
 - Multiple report formats with source, assembly, and instructions
- Measures all production applications

Resources

- caliper-help@cup.hp.com
 - Caliper “help” mailing list
 - All Caliper questions, bug reports, enhancement requests
- <http://hp.com/go/caliper>
 - Caliper external Web site
 - Downloads, documentation, tech notes
- <http://devresource.hp.com>
 - Developers’ Resource Web site
 - Training Webinars