HP-UX 11i Knowledge-on-Demand

HP technical Webcast series: software optimization

Technology for better business outcomes

© 2007 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice

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!

hp invent

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 invent

HP-UX Kernel Performance Tools

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

Technology for better business outcomes

© 2007 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice

Introducing today's speaker

Carol Petersen Muser is a senior system software engineer on the HP-UX Kernel Performance Team. She has worked on kernel tracing, kernel profiling, and related tools since 1997.

In this duration, she has improved the performance, quality, capabilities, ease of use and customer experience for these tools to provide faster troubleshooting and better insight into performance on HP-UX.

Agenda

- Overview
  - Features, Audience, Purpose
- User interface
  - ktracer, ktracedump, Caliper
- Sample output
- How does kernel tracing work?
- Case study
  - Troubleshoot a performance slowdown
- Additional features
- Conclusion
  - Availability and support

hp invent

Overview
Introduction to kernel tracing

- Purpose
  - To trace kernel procedure calls, parameters, timing
  - To observe, analyze & debug kernel behavior
- Advantages
  - functionality is **built in** into **every** kernel
    - PA and IPF, perf and debug flavor
  - Fast, powerful, flexible, extensible, insightful
- Available as ‘ktrace’ in the HP-UX Caliper product
  - http://hp.com/go/caliper

hp invent

Overview
kernel tracing attributes

- Works on all HP-UX releases and architectures
- No special build
- No performance cost when off
- Run time, boot up, live system, crashdump
- Superuser privilege
- Not intended for use in a production environment

Overview
kernel tracing features

ktracer traces

- CPU#, PID, TID, caller, callee, params, time, globals, ...
- Tens of thousands of functions. Any subset.
  - network, disk, filesystem, VM, PM, IO, SYNC, drivers
  - tcp_wput, escsi_strategy, kmem_alloc, setrq, clock_int
- kernel modules
  - **nfs, vxfs, inet, pm, io_forw, uipc, rng, vol, tcp, ip, td, ...**
- kernel libraries
  - **libigelan.a, librpc.a, libvm.a, libfs.a, ...**
- All processes or selected one

hp invent

Overview
ktracer audience

Ideal audience

- Wants Solaris DTrace or Linux FKT for HP-UX
- Knows Unix internals
- Has access to HP-UX kernel source code
- Develops software, DLKM's or drivers for HP-UX
- Is interested in HP-UX kernel behavior

Overview
How ktracer works

- User selects kernel functions to trace.
- ktracer places a trace point at the beginning of each selected function.
- *Each* and ***every*** time a selected function is called, one trace record is collected. Not sampling.
- Record traced function's address, arg0...arg3, caller. Also itimer, PID, TID, SpnD, SP, at every trace point
- Volumes of data collected, only recent is kept

hp invent

Overview
Trace records

Seq	Cpu	PID	TID	Caller	Function	AbsSec	ElUSec
100	1	7626	40674	bubbleup	ext_int	913.650	37.601
101	1	7626	40674	ext_int	clock_int	913.650	1.191
102	1	7626	40674	lv_parread	lv_begin	913.651	933.205
103	1	7626	40674	lv_begin	lv_startpv	913.651	17.893
104	1	7626	40674	lv_startpv	ia64dsk_ strategy	913.651	5.604
105	1	7626	40674	ia64dsk_ strategy	escsi_ strategy	913.651	50.542
106	1	7626	40674	swidle	idle	913.651	29.289
107	1	-1	-1	bubbleup	ext_int	913.660	8924.825
108	1	-1	-1	ext_int	clock_int	913.660	13.555

hp invent

Overview
Purpose of kernel tracing 1 of 2

- To observe, investigate, analyze, and learn kernel behavior
  - Which syscalls are made when by whom?
  - Which buffer, file, or socket is frequently used?
  - Does code of interest get called?
  - Which PIDs, CPUs and Callers acquire and hold a spinlock or read/write lock? How long is it held?
  - What are the implications of a kernel design decision?

hp invent

Overview
Purpose of kernel tracing 2 of 2

- To troubleshoot and debug problems
  - Identify the source of repeatable problems quickly
  - Functional defects
    - Are error handling routines invoked?
    - Is a function passed invalid parameters?
    - Who failed to release a resource?
    - Who shut off interrupts?
  - Performance issues
    - What is that unresponsive CPU doing?
    - Did a process get switched out during a critical region?
    - Does caller, callee sequence & timing show a path length issue?
    - Does contention on shared resources indicate a scalability problem?

hp invent

Agenda

- Overview
  - Features, Audience, Purpose
- User interface
  - ktracer, ktracedump, Caliper
- Sample output
- How does kernel tracing work?
- Case study
  - Troubleshoot a performance slowdown
- Additional features
- Conclusion
  - Availability and support

hp invent

ktracer user interface
All-in-one command line

- All-in-one command line: -g, -w
  - ktracer –**g** *nsec* > timed.trace.out
    - Traces the kernel for *nsec* seconds and produces report
  - ktracer –**w** ‘workload’ > workload.trace.out
    - Traces the kernel, runs the workload, produces report
- Both these options do all the needed steps, namely:
  1) **Allocate** and/or zero trace buffers (ktracer -A, -Z)
  2) Use existing **selec**ted **function list**, else select all funcs (-R)
  3) **Install** trace points and **begin** collecting traces (-I, -b)
  4) **Run** workload or sleep for *nsec* seconds
  5) **Stop** collecting traces, remove trace points (-h, -U)
  6) Invoke ktracedump to create a **report**

hp invent

ktracer user interface
Quick start options

ktracer -L and -R both do steps 1 (alloc) through 3 (begin) automatically.

ktracer –L

- ktracer "Lite" selects a **lightweight** set of key functions
- **Both arch:** *syscall, idle, swtch, resume_cleanup, assfail, panic*
- **Itanium:** *pre_hndlr, cr_itm_int, external_interrupt*
- **PA-RISC:** *trap, sampler, interrupt, up/mp_ext_interrupt*

ktracer –R

- ktracer "Reasonable" selects the **maximum** set of functions
- Includes tens of thousands. Excludes:
  - *Leaf functions on PA*
  - *Inlined function calls*
  - *Functions within DLKMs (Any beta testers in the audience?)*
  - *exclude list, irreplaceable instructions, bad unwind, $#.name*

hp invent

ktracer user interface
Selecting the traced function list

- ktracer –m *module_name* (11i v3/usr/conf/mod)
- ktracer –l *library_name* (11i v2/usr/conf/lib)
- ktracer –a *func_name -a func_addr*
  - Add func or explain why not
- ktracer -r func | -r 10
  - Remove func, or 10 most frequently traced funcs
- ktracer -z
  - zero the function list

hp invent

ktracer user interface Starting and Stopping

- Begin tracing: ktracer -b
- Halt (stop): ktracer -h
- Halt (stop) trace collection when a function is reached:
  - ktracer -S stop_func_name
  - ktracer -L and -R incorporate –S panic –S assfail
- Perf kernels start with ktracer off. Superuser controls that.
- Debug kernels start ktracer Lite during boot & leave it on.
- man ktracer(1M), ktracedump(1M)

hp invent

ktracer user interface
Simple usage scenario

A simple usage scenario

- ktracer –L
  - Turns on tracing of a few key functions relating to system calls, idle, interrupts and process switching
- Run a workload
  - E.g. a benchmark, test case, command or script
- ktracer –h
  - Turn off tracing
- ktracedump –D > kt.out
  - Produce a formatted report that displays the collected traces

hp invent

ktracer user interface
A more complicated example

- A more complicated example:

```
ktracer                       \
–L                            \    # trace "Lite" functions
–r external_interrupt         \    # except for 'external_interrupt'
–a btlan_isr                  \    # but add 'btlan_isr'
–m/usr/conf/mod/tcp           \    # plus all of tcp functions
–S dma_cleanup                \    # stop tracing if you get here
–b                                 # … now start tracing
```

Options are processed in order specified, left to right, order matters.

hp invent

ktracer user interface ktracedump command 1 of 2

ktracedump produces a formatted report from the trace data.

Fields of trace records are shown as columns of output

- ktracedump –D > kt.out
  – Report trace records from a live system
- ktracedump –m > kd.out
  – Report trace records from a crashdump
- ktracedump –FN > fn.list
  – Show the traced function list (-F), not the trace records (-N)
  – Think “FN” for FunctioN list

hp invent

ktracedump -H
Trace fields for column display

```
# ktracedump -H
Columns of printing which can be turned on or off:
     N  Name       Default(ON/OFF):Description
     1  H              ON      : Awk Parse Info
     2  SeqNum         OFF     : Sequence number
     3  ZSeq           ON      : Zero-based Sequence number
     4  Cpu            ON      : Processor Number
     5  PID            ON      : PID (process ID)
     6  TID            OFF     : TID (thread ID)
     7  SpnD           ON      : Spinlock Depth
     8  Caller         OFF     : Caller function
     9  COff           OFF     : Offset into caller function, in hex
    10  Function       ON      : Traced function (callee)
    11  AbsTime        OFF     : Interval Time Counter (ar.itc or cr16)
    12  AbsSec         OFF     : Absolute Time in Seconds.NanoSec
    13  ElTime         OFF     : Elapsed time since prev trace (iticks)
    14  ElUSec         ON      : Elapsed Time in microseconds.ns
    15  SP             OFF     : Stack pointer (kernel, ics)
```

ktracedump -H
Trace fields for column display (cont)

```
# ktracedump -H
Columns of printing which can be turned on or off:
    N  Name       Default(ON/OFF):Description
   16  PSR            ON        : Processor Status Register or Word (PSW)
   17  TPR            OFF       : Task Priority Register (TPR) or EIEM
   18  arg0           OFF       : Function Argument Value
   19  arg1           OFF       : Function Argument Value
   20  arg2           OFF       : Function Argument Value
   21  arg3           OFF       : Function Argument Value
   22  SymArg0        ON        : Symbolic Name for Function Argument
   23  SymArg1        OFF       : Symbolic Name for Function Argument
   24  SymArg2        OFF       : Symbolic Name for Function Argument
   25  SymArg3        OFF       : Symbolic Name for Function Argument
   26  global0        OFF       : Your variable
   27  global1        OFF       : Your variable
   28  global2        OFF       : Your variable
   29  global3        OFF       : Your variable
```


hp invent

ktracer user interface ktracedump command 2 of 2

- Request display of a column by name, optional format
  - ktracedump –J ColName%format e.g. -J AbsSec%16.9lf
- Exclude a column given its name or number
  - ktracedump –j ColNum e.g. –j PSR -j SpnD
- Display all columns for each trace record
  - ktracedump –A
- Sort the output by time rather than CPU
  - ktracedump –S AbsTime
- Use vim editor, www.vim.org, for large files, :set nowrap, syntax highlighting, multiple undo, multi-writer safety

hp invent

ktracer user interface
Caliper interface

- ktracer has been integrated into Caliper
- Use ktracer-args and ktracedump-args
- Example:

```
/opt/caliper/bin/caliper ktrace      \    # invoke ktracer
  --ktracer-args="-L –w test"        \    # run workload "test"
  --ktracedump-args="-D –J arg1"     \    # format the output
  -o ktrace.out                      \    # redirect output
```


hp invent

ktracer user interface
Simple usage scenario for learning

ktracer –L –w workload > ktrace.out

- -L turns on tracing of a few key functions relating to system calls, idle, interrupts and process switching
- -w will start tracing, execute “workload”, stop tracing
- When the workload completes, invoke ktracedump to:
  - Extract the trace records and function list from the kernel
  - Interpret and format the trace data
  - Write the report to file “ktrace.out”
- The next section shows the ktrace.out content in detail.

Agenda

- Overview
  - Features, Audience, Purpose
- User interface
  - ktracer, ktracedump, Caliper
- Sample output
  - file header, trace header, columns, trace records
- How does kernel tracing work?
- Case study
  - Troubleshoot a performance slowdown
- Additional features
- Conclusion
  - Availability and support

ktracedump sample output 1 of 14: ktrace.out file header

```
ktracer: INFO: starting workload 'workload' at Mon Jul 9 15:12:51 2007
ktracer: INFO: finished workload 'workload' at Mon Jul 9 15:12:51 2007
Executing: ktracedump -D

ktracedump 3.4.2 2007/04/02 20:00
ktracedump compiled on Jun 12 2007 10:31:56
ktracedump invoked on Mon Jul 9 15:12:54 2007

Kernel release_version = @(#) $Revision: vmunix:     jazz @
20070618.12:13:06PDT; jmkvw -proj kern1 -RW -c Task:
r11.31(R11.31_BL2007_0612); FLAVOR=perf
Kernel linkstamp = Mon Jun 18 12:16:14 PDT 2007
```

ktracedump sample output 2 of 14: file header

```
AbsTime and ElTime are in iticks. itick/sec=1.000 GHz
nanosec/itick=1.000
AbsTime is adjusted by tod_info.offset_correction

The circular trace buffers hold 2048 traces per CPU.
2 CPUs are traced, making 4096 traces total. nmpinfo=2
The traces consume 0.6 Mb (160 pages) of memory.
Q total # of traces gathered = 12

S Traces are sorted by Cpu# primary, AbsTime secondary
```


hp invent

ktracedump sample output 3 of 14: trace header

```
C    column#     Header  - Description                              Format
C    -------     ------  - ------------                             ------
F          1          H  - Awk Parse Info                             %1s
F          2       ZSeq  - Zero-based Sequence number               %611u
F          3        Cpu  - Processor Number                           %3u
F          4        PID  - PID (process ID)                           %5d
F          5       SpnD  - Spinlock Depth                             %4d
F          6   Function  - Traced function (callee)                 %-20s
F          7     ElUSec  - Elapsed Time in microseconds.ns        %10.31f
F          8        PSR  - PSR pp/ic/pk/dt/i = pCPDI or _____         %5s
F          9    SymArg0  - Symbolic Name for Function Argument      %-18s
```

ktracedump sample output
4 of 14: trace header and trace records

```
C Spu 1; 12 traces; The columns mean:
C      column#        Header  -  Description                                    -  Format
C      -------        ------  -  -----------------------------------------------  -  ------
F            1             H  -  Awk Parse Info                                 -  %1s
F            2          ZSeq  -  Zero-based Sequence number                     -  %611u
F            3           Cpu  -  Processor Number                               -  %3u
F            4           PID  -  PID (process ID)                               -  %5d
F            5          SpnD  -  Spinlock Depth                                 -  %4d
F            6      Function  -  Traced function (callee)                       -  %-20s
F            7        ElUSec  -  Elapsed Time in microseconds.ns                -  %10.31f
F            8           PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off       -  %5s
F            9       SymArg0  -  Symbolic Name for Function Argument            -  %-18s
#
H    ZSeq   Cpu       PID   SpnD  Function                      ElUSec   PSR       SymArg0
T       0     1     28231      0  syscall                        0.000   pCPDI     siginhibit
T       1     1     28231      0  syscall                       21.719   pCPDI     sigenable
T       2     1     28231      0  syscall                        1.621   pCPDI     kill
T       3     1     28231      0  syscall                        7.956   pCPDI     siginhibit
T       4     1     28231      0  syscall                        5.267   pCPDI     sigenable
T       5     1     28231      0  syscall                        2.150   pCPDI     wait
T       6     1     28231      1  resume_cleanup                 5.190   pCPD_     0x10
T       7     1     28234      1  swtch                        436.322   pCPDI     0xe00000013c076b80
T       8     1     28234      1  resume_cleanup                 3.677   pCPD_     0x10
T       9     1     28231      0  syscall                       31.407   pCPDI     time
T      10     1     28231      0  syscall                       46.232   pCPDI     toolbox
T      11     1     28231      0  syscall                        1.907   pCPDI     toolbox
```

ktracedump sample output 5 of 14: ZSeq column

```
C Spu 1; 12 traces; The columns mean:
C       column#        Header  -  Description                                        -  Format
C       -------        ------  -  ---------------------------------------------------  -  ------
F             1             H  -  Awk Parse Info                                     -  %1s
F             2          ZSeq  -  Zero-based Sequence number                         -  %611u
F             3           Cpu  -  Processor Number                                   -  %3u
F             4           PID  -  PID (process ID)                                   -  %5d
F             5          SpnD  -  Spinlock Depth                                     -  %4d
F             6      Function  -  Traced function (callee)                           -  %-20s
F             7        ElUSec  -  Elapsed Time in microseconds.ns                    -  %10.31f
F             8           PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off           -  %5s
F             9       SymArg0  -  Symbolic Name for Function Argument                -  %-18s
#
H    ZSeq  Cpu     PID   SpnD  Function                     ElUSec  PSR      SymArg0
T       0    1   28231      0  syscall                       0.000  pCPDI    siginhibit
T       1    1   28231      0  syscall                      21.719  pCPDI    sigenable
T       2    1   28231      0  syscall                       1.621  pCPDI    kill
T       3    1   28231      0  syscall                       7.956  pCPDI    siginhibit
T       4    1   28231      0  syscall                       5.267  pCPDI    sigenable
T       5    1   28231      0  syscall                       2.150  pCPDI    wait
T       6    1   28231      1  resume_cleanup                5.190  pCPD_    0x10
T       7    1   28234      1  swtch                       436.322  pCPDI    0xe00000013c076b80
T       8    1   28234      1  resume_cleanup                3.677  pCPD_    0x10
T       9    1   28231      0  syscall                      31.407  pCPDI    time
T      10    1   28231      0  syscall                      46.232  pCPDI    toolbox
T      11    1   28231      0  syscall                       1.907  pCPDI    toolbox
```

ktracedump sample output 6 of 14: CPU column

```
C Spu 1; 12 traces; The columns mean:
C       column#         Header  -  Description                                          -  Format
C       -------         ------  -  --------------------------------------------------  -  ------
F             1              H  -  Awk Parse Info                                       -  %1s
F             2           ZSeq  -  Zero-based Sequence number                           -  %611u
F             3            Cpu  -  Processor Number                                     -  %3u
F             4            PID  -  PID (process ID)                                     -  %5d
F             5           SpnD  -  Spinlock Depth                                       -  %4d
F             6       Function  -  Traced function (callee)                             -  %-20s
F             7         ElUSec  -  Elapsed Time in microseconds.ns                      -  %10.31f
F             8            PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off             -  %5s
F             9        SymArg0  -  Symbolic Name for Function Argument                  -  %-18s
#
H     ZSeq   Cpu       PID   SpnD  Function                      ElUSec  PSR       SymArg0
T        0     1     28231      0  syscall                        0.000  pCPDI     siginhibit
T        1     1     28231      0  syscall                       21.719  pCPDI     sigenable
T        2     1     28231      0  syscall                        1.621  pCPDI     kill
T        3     1     28231      0  syscall                        7.956  pCPDI     siginhibit
T        4     1     28231      0  syscall                        5.267  pCPDI     sigenable
T        5     1     28231      0  syscall                        2.150  pCPDI     wait
T        6     1     28231      1  resume_cleanup                 5.190  pCPD_     0x10
T        7     1     28234      1  swtch                        436.322  pCPDI     0xe00000013c076b80
T        8     1     28234      1  resume_cleanup                 3.677  pCPD_     0x10
T        9     1     28231      0  syscall                       31.407  pCPDI     time
T       10     1     28231      0  syscall                       46.232  pCPDI     toolbox
T       11     1     28231      0  syscall                        1.907  pCPDI     toolbox
```

ktracedump sample output 7 of 14: PID column

```
C Spu 1; 12 traces; The columns mean:
C       column#        Header  -  Description                                        -  Format
C       -------        ------  -  --------------------------------------------------  -  ------
F             1             H  -  Awk Parse Info                                      -  %1s
F             2          ZSeq  -  Zero-based Sequence number                          -  %611u
F             3           Cpu  -  Processor Number                                    -  %3u
F             4           PID  -  PID (process ID)                                    -  %5d
F             5          SpnD  -  Spinlock Depth                                      -  %4d
F             6      Function  -  Traced function (callee)                            -  %-20s
F             7        ElUSec  -  Elapsed Time in microseconds.ns                     -  %10.31f
F             8           PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off            -  %5s
F             9       SymArg0  -  Symbolic Name for Function Argument                 -  %-18s
#
H     ZSeq   Cpu       PID   SpnD  Function                ElUSec   PSR        SymArg0
T        0     1     28231      0  syscall                  0.000   pCPDI      siginhibit
T        1     1     28231      0  syscall                 21.719   pCPDI      sigenable
T        2     1     28231      0  syscall                  1.621   pCPDI      kill
T        3     1     28231      0  syscall                  7.956   pCPDI      siginhibit
T        4     1     28231      0  syscall                  5.267   pCPDI      sigenable
T        5     1     28231      0  syscall                  2.150   pCPDI      wait
T        6     1     28231      1  resume_cleanup           5.190   pCPD_      0x10
T        7     1     28234      1  swtch                  436.322   pCPDI      0xe00000013c076b80
T        8     1     28234      1  resume_cleanup           3.677   pCPD_      0x10
T        9     1     28231      0  syscall                 31.407   pCPDI      time
T       10     1     28231      0  syscall                 46.232   pCPDI      toolbox
T       11     1     28231      0  syscall                  1.907   pCPDI      toolbox
```

ktracedump sample output 8 of 14: SpnD column

```
C Spu 1; 12 traces; The columns mean:
C       column#         Header  -  Description                                        -  Format
C       -------         ------  -  --------------------------------------------------  -  ------
F             1              H  -  Awk Parse Info                                     -  %1s
F             2           ZSeq  -  Zero-based Sequence number                         -  %611u
F             3            Cpu  -  Processor Number                                   -  %3u
F             4            PID  -  PID (process ID)                                   -  %5d
F             5           SpnD  -  Spinlock Depth                                     -  %4d
F             6       Function  -  Traced function (callee)                           -  %-20s
F             7         ElUSec  -  Elapsed Time in microseconds.ns                    -  %10.31f
F             8            PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off           -  %5s
F             9        SymArg0  -  Symbolic Name for Function Argument                -  %-18s
#
H    ZSeq   Cpu       PID   SpnD  Function                     ElUSec  PSR         SymArg0
T       0     1     28231      0  syscall                       0.000  pCPDI       siginhibit
T       1     1     28231      0  syscall                      21.719  pCPDI       sigenable
T       2     1     28231      0  syscall                       1.621  pCPDI       kill
T       3     1     28231      0  syscall                       7.956  pCPDI       siginhibit
T       4     1     28231      0  syscall                       5.267  pCPDI       sigenable
T       5     1     28231      0  syscall                       2.150  pCPDI       wait
T       6     1     28231      1  resume_cleanup                5.190  pCPD_       0x10
T       7     1     28234      1  swtch                       436.322  pCPDI       0xe00000013c076b80
T       8     1     28234      1  resume_cleanup                3.677  pCPD_       0x10
T       9     1     28231      0  syscall                      31.407  pCPDI       time
T      10     1     28231      0  syscall                      46.232  pCPDI       toolbox
T      11     1     28231      0  syscall                       1.907  pCPDI       toolbox
```

ktracedump sample output
9 of 14: Function column

```
C Spu 1; 12 traces; The columns mean:
C       column#        Header  -  Description                                         -  Format
C       -------        ------  -  --------------------------------------------------  -  ------
F             1             H  -  Awk Parse Info                                      -  %1s
F             2          ZSeq  -  Zero-based Sequence number                          -  %611u
F             3           Cpu  -  Processor Number                                    -  %3u
F             4           PID  -  PID (process ID)                                    -  %5d
F             5          SpnD  -  Spinlock Depth                                      -  %4d
F             6      Function  -  Traced function (callee)                            -  %-20s
F             7        ElUSec  -  Elapsed Time in microseconds.ns                     -  %10.31f
F             8           PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off            -  %5s
F             9       SymArg0  -  Symbolic Name for Function Argument                 -  %-18s
#
H    ZSeq   Cpu      PID   SpnD  Function                     ElUSec  PSR     SymArg0
T       0     1    28231      0  syscall                       0.000  pCPDI   siginhibit
T       1     1    28231      0  syscall                      21.719  pCPDI   sigenable
T       2     1    28231      0  syscall                       1.621  pCPDI   kill
T       3     1    28231      0  syscall                       7.956  pCPDI   siginhibit
T       4     1    28231      0  syscall                       5.267  pCPDI   sigenable
T       5     1    28231      0  syscall                       2.150  pCPDI   wait
T       6     1    28231      1  resume_cleanup                5.190  pCPD_   0x10
T       7     1    28234      1  swtch                       436.322  pCPDI   0xe00000013c076b80
T       8     1    28234      1  resume_cleanup                3.677  pCPD_   0x10
T       9     1    28231      0  syscall                      31.407  pCPDI   time
T      10     1    28231      0  syscall                      46.232  pCPDI   toolbox
T      11     1    28231      0  syscall                       1.907  pCPDI   toolbox
```

ktracedump sample output
10 of 14: ElUSec column

```
C Spu 1; 12 traces; The columns mean:
C       column#        Header  -  Description                                          -  Format
C       -------        ------  -  -------------------------------------------------    -  ------
F             1             H  -  Awk Parse Info                                       -  %1s
F             2          ZSeq  -  Zero-based Sequence number                           -  %611u
F             3           Cpu  -  Processor Number                                     -  %3u
F             4           PID  -  PID (process ID)                                     -  %5d
F             5          SpnD  -  Spinlock Depth                                       -  %4d
F             6      Function  -  Traced function (callee)                             -  %-20s
F             7        ElUSec  -  Elapsed microsec since prev trace                    -  %10.31f
F             8           PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off             -  %5s
F             9       SymArg0  -  Symbolic Name for Function Argument                  -  %-18s
#
H    ZSeq  Cpu      PID  SpnD  Function                ElUSec  PSR     SymArg0
T       0    1    28231     0  syscall                  0.000  pCPDI   siginhibit
T       1    1    28231     0  syscall                 21.719  pCPDI   sigenable
T       2    1    28231     0  syscall                  1.621  pCPDI   kill
T       3    1    28231     0  syscall                  7.956  pCPDI   siginhibit
T       4    1    28231     0  syscall                  5.267  pCPDI   sigenable
T       5    1    28231     0  syscall                  2.150  pCPDI   wait
T       6    1    28231     1  resume_cleanup           5.190  pCPD_   0x10
T       7    1    28234     1  swtch                  436.322  pCPDI   0xe00000013c076b80
T       8    1    28234     1  resume_cleanup           3.677  pCPD_   0x10
T       9    1    28231     0  syscall                 31.407  pCPDI   time
T      10    1    28231     0  syscall                 46.232  pCPDI   toolbox
T      11    1    28231     0  syscall                  1.907  pCPDI   toolbox
```

ktracedump sample output 11 of 14: PSR column

```
C Spu 1; 12 traces; The columns mean:
C      column#      Header  -  Description                                       -  Format
C      -------      ------  -  ------------------------------------------------  -  ------
F            1           H  -  Awk Parse Info                                    -  %1s
F            2        ZSeq  -  Zero-based Sequence number                        -  %611u
F            3         Cpu  -  Processor Number                                  -  %3u
F            4         PID  -  PID (process ID)                                  -  %5d
F            5        SpnD  -  Spinlock Depth                                    -  %4d
F            6    Function  -  Traced function (callee)                          -  %-20s
F            7      ElUSec  -  Elapsed Time in microseconds.ns                   -  %10.31f
F            8         PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off          -  %5s
F            9     SymArg0  -  Symbolic Name for Function Argument               -  %-18s
#
H   ZSeq  Cpu     PID  SpnD  Function                     ElUSec  PSR    SymArg0
T      0    1   28231     0  syscall                       0.000  pCPDI  siginhibit
T      1    1   28231     0  syscall                      21.719  pCPDI  sigenable
T      2    1   28231     0  syscall                       1.621  pCPDI  kill
T      3    1   28231     0  syscall                       7.956  pCPDI  siginhibit
T      4    1   28231     0  syscall                       5.267  pCPDI  sigenable
T      5    1   28231     0  syscall                       2.150  pCPDI  wait
T      6    1   28231     1  resume_cleanup                5.190  pCPD_  0x10
T      7    1   28234     1  swtch                       436.322  pCPDI  0xe00000013c076b80
T      8    1   28234     1  resume_cleanup                3.677  pCPD_  0x10
T      9    1   28231     0  syscall                      31.407  pCPDI  time
T     10    1   28231     0  syscall                      46.232  pCPDI  toolbox
T     11    1   28231     0  syscall                       1.907  pCPDI  toolbox
```

ktracedump sample output 12 of 14: SymArg0 column

```
C Spu 1; 12 traces; The columns mean:
C       column#         Header  -  Description                                         -  Format
C       -------         ------  -  --------------------------------------------------  -  ------
F             1              H  -  Awk Parse Info                                      -  %1s
F             2           ZSeq  -  Zero-based Sequence number                          -  %611u
F             3            Cpu  -  Processor Number                                    -  %3u
F             4            PID  -  PID (process ID)                                    -  %5d
F             5           SpnD  -  Spinlock Depth                                      -  %4d
F             6       Function  -  Traced function (callee)                            -  %-20s
F             7         ElUSec  -  Elapsed Time in microseconds.ns                     -  %10.31f
F             8            PSR  -  PSR pp/ic/pk/dt/i = pCPDI when on, _ off            -  %5s
F             9        SymArg0  -  Symbolic Name for Function Argument                 -  %-18s
#
H    ZSeq   Cpu      PID   SpnD  Function                     ElUSec   PSR      SymArg0
T       0     1    28231      0  syscall                       0.000   pCPDI    siginhibit
T       1     1    28231      0  syscall                      21.719   pCPDI    sigenable
T       2     1    28231      0  syscall                       1.621   pCPDI    kill
T       3     1    28231      0  syscall                       7.956   pCPDI    siginhibit
T       4     1    28231      0  syscall                       5.267   pCPDI    sigenable
T       5     1    28231      0  syscall                       2.150   pCPDI    wait
T       6     1    28231      1  resume_cleanup                5.190   pCPD_    0x10
T       7     1    28234      1  swtch                       436.322   pCPDI    0xe00000013c076b80
T       8     1    28234      1  resume_cleanup                3.677   pCPD_    0x10
T       9     1    28231      0  syscall                      31.407   pCPDI    time
T      10     1    28231      0  syscall                      46.232   pCPDI    toolbox
T      11     1    28231      0  syscall                       1.907   pCPDI    toolbox
```

ktracedump sample output 13 of 14: trace record analysis

H	ZSeq	Cpu	PID	SpnD	Function	ElUSec	PSR	SymArg0
T	0	1	28231	0	syscall	0.000	pCPDI	siginhibit
T	1	1	28231	0	syscall	21.719	pCPDI	sigenable
T	2	1	28231	0	syscall	1.621	pCPDI	kill
T	3	1	28231	0	syscall	7.956	pCPDI	siginhibit
T	4	1	28231	0	syscall	5.267	pCPDI	sigenable
T	5	1	28231	0	syscall	2.150	pCPDI	wait
T	6	1	28231	1	resume_cleanup	5.190	pCPD_	0x10
T	7	1	28234	1	swtch	436.322	pCPDI	0xe00000013c076b80
T	8	1	28234	1	resume_cleanup	3.677	pCPD_	0x10
T	9	1	28231	0	syscall	31.407	pCPDI	time
T	10	1	28231	0	syscall	46.232	pCPDI	toolbox
T	11	1	28231	0	syscall	1.907	pCPDI	toolbox

Function histogram:

T for Trace record, $6 for Function (col 6)

```
% egrep "^T " ktrace.out | awk '{print $6}' | sort | uniq -c
2 resume_cleanup
1 swtch
9 syscall
```


hp invent

ktracedump sample output 14 of 14: trace record analysis

H	ZSeq	Cpu	PID	SpnD	Function	ElUSec	PSR	SymArg0
T	0	1	28231	0	syscall	0.000	pCPDI	siginhibit
T	1	1	28231	0	syscall	21.719	pCPDI	sigenable
T	2	1	28231	0	syscall	1.621	pCPDI	kill
T	3	1	28231	0	syscall	7.956	pCPDI	siginhibit
T	4	1	28231	0	syscall	5.267	pCPDI	sigenable
T	5	1	28231	0	syscall	2.150	pCPDI	wait
T	**6**	**1**	**28231**	**1**	**resume_cleanup**	**5.190**	pCPD_	0x10
T	**7**	**1**	**28234**	**1**	**swtch**	**436.322**	pCPDI	0xe00000013c076b80
T	8	1	28234	1	resume_cleanup	3.677	pCPD_	0x10
T	9	1	28231	0	syscall	31.407	pCPDI	time
T	10	1	28231	0	syscall	46.232	pCPDI	toolbox
T	11	1	28231	0	syscall	1.907	pCPDI	toolbox

The largest ElUSec, 436.322, shows how many microseconds elapsed between trace 6 (call to resume_cleanup) and trace 7 (call to swtch)

Ref Project Mercury (hg) to reduce inopportune process switching

hp invent

Agenda

- Overview
  - Features, Audience, Purpose
- User interface
  - ktracer, ktracedump, Caliper
- Sample output
- How does kernel tracing work?
  - Trace buffers, trace points, performance
- Case study
  - Troubleshoot a performance slowdown
- Additional features
- Conclusion
  - Availability and support

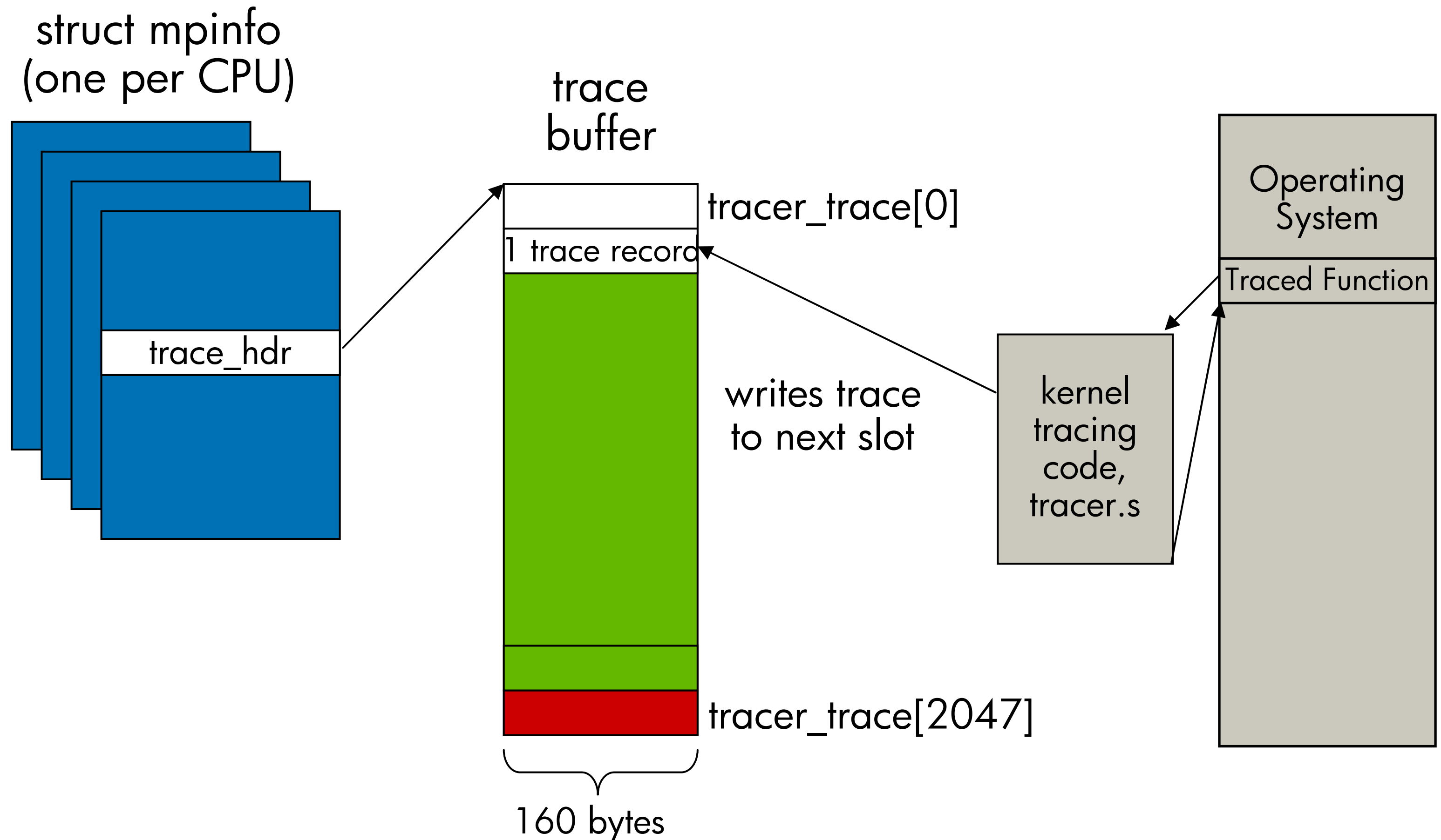
hp invent

How does kernel tracing work? Trace Buffers

- Each CPU has its own trace buffer, allocated from kernel memory.
- Each trace buffer can hold up to 2048 trace records by default.
- After 2048 procedure calls are traced, the trace buffer becomes full.
- Trace records are called "traces" for short.
- Use ‘ktracer –A *ntraces*’ to adjust the # of traces each CPU can store
- System-wide safeguard of 6 million traces (1GB of space).
- adb max_ntraces to allow trace buffers to consume as much kernel memory as requested and available.
- ktracedump cannot extract old trace records if the trace buffers are cleared (ktracer -Z), freed (ktracer -f), or overwritten by further activity.

invent

How does kernel tracing work?
Trace Buffers



How does kernel tracing work? Trace Buffer Management

- Manage trace buffers as **Circular** (default) or **Linear**

  Circular: when a trace buffer becomes full, ktracer wraps around and replaces the oldest trace on that CPU with the latest trace. The oldest traces will be lost when overwritten.

  Linear: when a trace buffer becomes full, no more traces are collected for that CPU. The newest traces will be lost.

  When trace records are lost, we call it trace buffer overflow.

  Each CPU fills its trace buffer at the rate that traced procedure calls are made. This rate may vary from one CPU to another.

  Be aware that trace record loss will cause the oldest traces if circular, or the newest traces if linear, to be missing from *each CPU* that was busy enough tracing to overflow the size of its trace buffer.

How does kernel tracing work? Trace Timing

- The timestamp in each trace record is taken from the CPU’s interval timer, and adjusted by the offset vs. monarch.
- 'ktracedump -S AbsSec' sorts traces by timestamp across all CPUs.
  - excessive clock drift will make the sort order incorrect
  - trace record loss on a per-CPU basis becomes apparent.

To avoid trace buffer overflow:

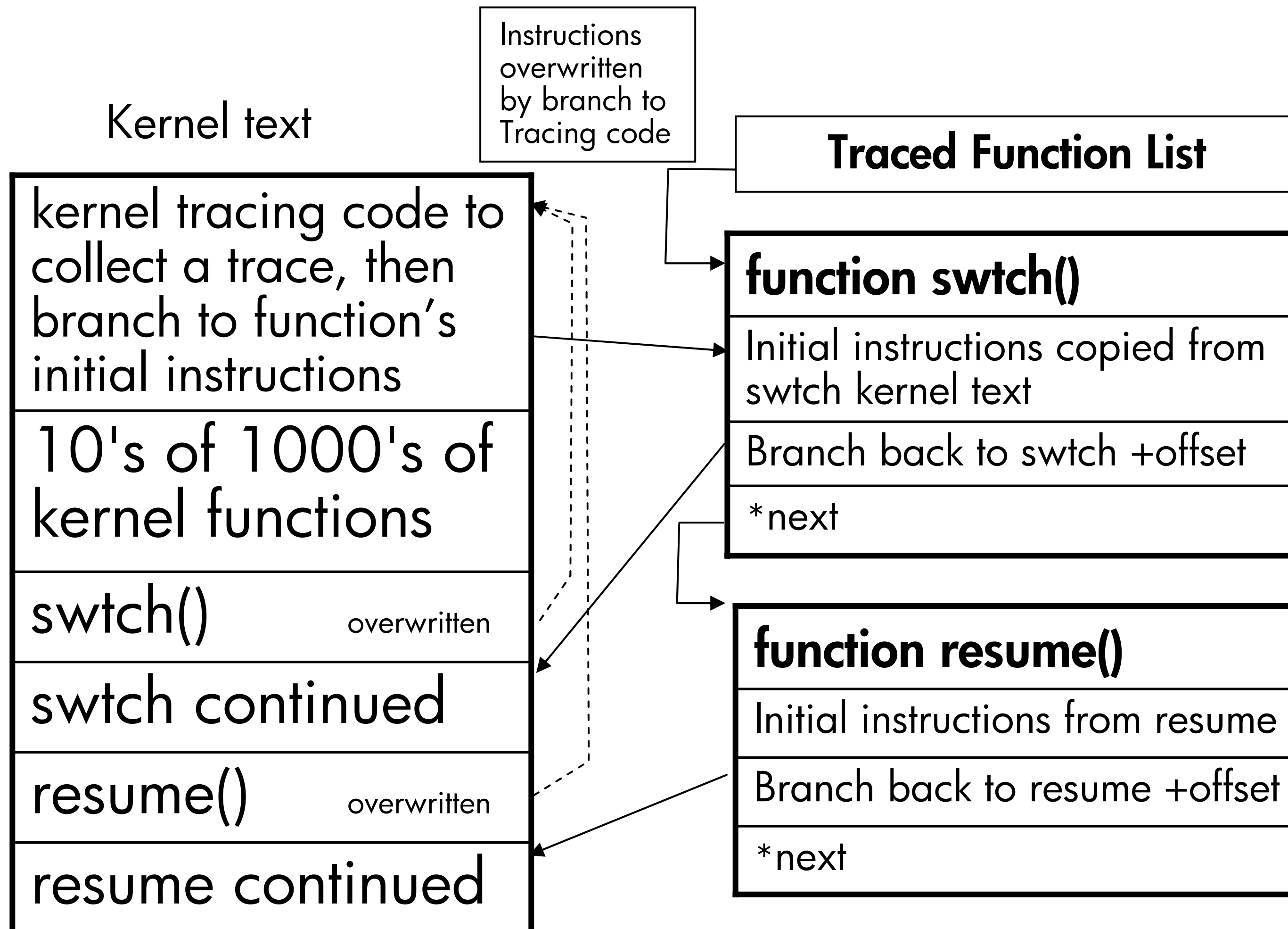
1. Increase the # of trace records each CPU can store (ktracer -A)
2. Stop tracing the most frequently called procedures (ktracer -r,-z,-a)
3. Trace for a shorter timeframe
4. Trace only one process (ktracer -p pid)

How does kernel tracing work?
Trace Points

- A trace point is a place where the kernel will branch to tracing code, collect a trace record, then branch back to resume normal path flow.
- ktracer –b installs a trace point at each selected function and begins data collection.
- Trace points exist at the *beginning* of selected procedures only.
- Creating a trace point involves on-the-fly modification of the kernel text image in memory
- ktracer –h halts data collection and removes all trace points, restoring original performance.
- Customize the set of trace points dynamically for each investigation

hp invent

How does kernel tracing work? Trace Points in Kernel Text



How does kernel tracing work? Performance overhead

- The performance cost to run ktracer depends directly on the frequency of calls to functions that are traced.
- Optimally, the time it takes to capture each trace is
  - 110 cycles on PA
  - 90 cycles on IPF **10 times faster than 11i v2 IPF**
- It is faster to trace hundreds of functions that are infrequently called than to trace one single function that is intensively called.
- Suppose clock_int() is called 100 times *per second* on a PA 1.0GHz box. To trace clock_int for 1 sec would cost:
  (100 traces) * (110 cyc/trace)/(1G cyc/sec) = 11 usec
  which is 0.0011 percent of the 1 sec interval
- Suppose intr_strobe_clear_idle() is called 500 times *per millisecond* on a 1.6GHz Itanium box. To trace it for 1 ms would cost:
  (500 traces) * (90 cyc/trace)/(1.6G cyc/sec) = 28 usec
  which is 2.8% of the 1ms interval

hp invent

Agenda

- Overview
  - Features, Audience, Purpose
- User interface
  - ktracer, ktracedump, Caliper
- Sample output
- How does kernel tracing work?
- Case study
  - Troubleshoot a performance slowdown
- Additional features
- Conclusion
  - Availability and support

hp invent

ktracer case study
Troubleshoot a Performance Slowdown

- Scenario:
  - After upgrading OS, benchmark performance regressed 18%. Same machine, benchmark, disks, & database. Diff root disk.
  - Task: Figure out why. Who is the code owner to report it to?
  - Symptoms:
    - time(1M) indicates 627 seconds real time vs 557 prev to run small benchmark. 28 sec more in user, 3 sec more in sys
    - sar -d indicates 100x longer disk avwait and avqueue
    - spinwatcher identifies more user time in sas, more kernel time in idle. Both prev and new runs spend the most time in processes sas and vxfsd, and the most kernel cpu time in vx_inode_free_list
  - What changed in the OS? VXFS 3.5 to 4.1, UFC, disk paths,…
  - Who should own this? Where exactly is the slowdown?

hp invent

ktracer case study
Troubleshoot a Performance Slowdown

- ktracer to the rescue
  - Trace the kernel activity of the vxfsd process: ktracer -p 59
  - Select all the functions in the vxfs module: ktracer -m vxfs
  - Run ktracer -b as sas begins.
  - Run ktracer -h as it ends.
  - Dump the traces and analyze them. ktracedump -D > out
  - Discover lots of activity around functions named vx_recsmp_rangelock and vx_recsmp_rangeunlock.
  - Trace just these 2 locking functions and revert to tracing all processes to fully see rwlock contention.
    ktracer -z -a vx_recsmp_rangelock -a vx_recsmp_rangeunlock -p 0

hp invent

ktracer case study
Troubleshoot a Performance Slowdown

- Run ktracer iteratively to gain insight
  - Start ktracer once again when the last phase of sas begins ktracer -Z -b, and run ktracer -h before it ends.
  - Invoke ktracedump -J Caller -J AbsSec -J arg0 -F to add trace fields that are interesting. -F shows the 2 functions being traced
  - Write a perl script that parses ktracedump output and calculates recsmp range lock hold duration based on AbsSec diff, using arg0 to identify unique lock addrs.
  - Some range unlocks mid-run have no matching lock.
  - Search nm output, and run ktracer -a vx_recsmp_rangetrylock so that all vx_recsmp_range*lock functions are traced.

hp invent

ktracer case study
Troubleshoot a Performance Slowdown

- Isolate the performance regression
  - Capture the trace data again during last phase of sas.
  - Accurately identify long hold times for vx_recsmp_rangelock.
  - Map lock arg0 into an vx_inode and get its path name. This works only for names not pushed out of the DNLC.
  - Send ktracer data and problem report to vxfs owner.
  - Resolution: http://support.veritas.com/docs/290636.htm
  - Run vxtunefs to increase max_diskq from 1MB to 2048MB

hp invent

Agenda

- Overview
  - Features, Audience, Purpose
- User interface
  - ktracer, ktracedump, Caliper
- Sample output
- How does kernel tracing work?
- Case study
  - Troubleshoot a performance slowdown
- Additional features
- Conclusion
  - Availability and support

hp invent

Additional Features
Inserting trace points anywhere

- If you own kernel source code, you can add a function call such as kt_dbg(ulong_t a,b,c,d) to it and trace the new function call.
  For example: kt_dbg (__LINE__, lbolt, u.u_error, buf);
- **ktracer -a kt_dbg -D '-a' -w workload > kt.out**
- To see the values of __LINE__, lbolt, u.u_error, and buf each time kt_dbg was called during workload, look in columns arg0, arg1, arg2 and arg3 of kt.out.
- You can use kt_dbg() like a printf of 0 to 4 scalar variables, with theses benefits over printf:
  - lower runtime performance cost
  - doesn't flood console, searchable (unlike uprintf)
  - column aligned output, adjustable by -Jargn%fmt

If **ktracer -a foo** indicates it can't trace leaf functions, add a kt_dbg call to foo so it's not a leaf function anymore.

Additional Features
Embedded calls

- An API exists to allow control over when ktracer is activated
- Embed ktracer calls into the application
- Frequently used to halt data collection when a user-defined condition has been reached
  - This prevents losing interesting traces due to buffer overflow
- Use to enable tracing only during the important part of the application
  - Reduces overhead during the rest of the run

hp invent

Agenda

- Overview
  - Features, Audience, Purpose
- User interface
  - ktracer, ktracedump, Caliper
- Sample output
- How does kernel tracing work?
- Case study
  - Troubleshoot a performance slowdown
- Additional features
- Conclusion
  - Availability and support

Conclusion Summary

- HP-UX kernel tracing is fast and built-in
- kernel tracing helps engineers
  - to learn and understand HP-UX kernel behavior, and
  - to troubleshoot functional and performance problems by capturing data when the kernel calls functions.
- Quick start and do-it-all options make ktracer easy to invoke.
- Unix internals knowledge and HP-UX source access are best to make sense of trace output.
- Procedure selection is strategic to problem solving, trace buffer use, and reduced performance overhead.

hp invent

Conclusion
Related products and services

If you're interested in this topic, you may also like

- kprof – kernel profiling
- spinwatcher – kernel spinlock contention
- an incredibly fast, defect-free OS
- a winning lottery ticket

hp invent

Conclusion
Where to find additional information

- Caliper documentation and download
  - http://hp.com/go/caliper
- About ktracer and kernel profiling
  - email: Carol.Muser @ hp.com
- General
  - google.com

hp invent